Efficient, Edge-Aware, Combined Color Quantization and Dithering

Hao-Zhi Huang, Kun Xu, *Member, IEEE,* Ralph R. Martin, Fei-Yue Huang, and Shi-Min Hu, *Member, IEEE,*

***Abstract*—In this paper we present a novel algorithm to simultaneously accomplish color quantization and dithering of images. This is achieved by minimizing a perception-based cost function which considers pixel-wise differences between filtered versions of the quantized image and the input image. We use edge aware filters in defining the cost function to avoid mixing colors on opposite sides of an edge. The importance of each pixel is weighted according to its saliency. To rapidly minimize the cost function, we use a modified multi-scale iterative conditional mode (ICM) algorithm which updates one pixel a time while keeping other pixels unchanged. As ICM is a local method, careful initialization is required to prevent termination at a local minimum far from the global one. To address this problem, we initialize ICM with a palette generated by a modified median-cut method. Compared to previous approaches, our method can produce high quality results with fewer visual artifacts but also requires significantly less computational effort.**

***Index Terms*—Color quantization, dithering, optimization-based image processing.**

I. Introduction

COLOR quantization is a technique that reduces the number of unique colors used in an image to a small number such as 256 or fewer, while preserving visual similarity to the original input image. It has applications in displaying and printing images on hardware devices which only support a small number of colors. It can also be used as a fundamental preprocessing step in lossy compression, fast image manipulation, image analysis and many other tasks.

A quantized image is represented by a *color palette*, which stores a set of unique colors, and a *pixel map*, which records the assignment of each pixel to one of the palette colors. A color quantization process typically involves two steps: color palette construction, and pixel map assignment [1]. Since natural images may contain hundreds of thousands of different colors, obtaining a visually pleasing quantized representation using a small color palette is a difficult problem.

Earlier works on color quantization are mainly based on clustering. Some of these methods use specific data structures to divide the color space, such as median-cut [2], or octrees [3]. Others use iterative clustering methods, such as k-means [1] or self-organizing maps [4], to organize all colors into a limited number of clusters. All colors in the same cluster are then represented using the same color, resulting in a quantized representation. However, they only consider whether the color of each pixel is close to its cluster center. Given the limited number of colors, and the fact that such methods ignore spatial contexts, a severe type of artifact arises in the quantized image in areas of smooth color gradients, in the form of *false edges*, as are clearly visible in the sky in Fig. 1b, for example. To reduce such artifacts, a subsequent *dithering* step [5] is typically employed after quantization, as the human visual system tends to perceive regions with high-frequency spatial color changes as a homogeneous color. Dithering distributes quantization errors into neighboring pixels, helping to hide the false edges. The improvements provided by dithering can be seen, for example, in Fig. 1c. However, there is an obvious conflict between these two steps. While quantization tries to reduce the average distance between each pixel and its cluster center, dithering increases this average distance in an attempt to provide smoothly varying colors. Performing these two steps independently will obviously lead to a suboptimal result.

Thus, instead of using a sequential approach, Puzicha [6] proposed the concept of *spatial quantization*, which simultaneously performs quantization and dithering by minimizing a quantization error. To simultaneously incorporate dithering into the minimization problem, the quantization error is defined as pixel-wise differences between a Gaussian-filtered quantized image and the Gaussian-filtered input image. Minimization is performed by deterministic annealing (DA) and an iterative conditional mode algorithm (ICM) using a multi-scale framework. The DA method, which is slow but more accurate, is used first for minimization. When the annealing temperature is close to zero, the method switches to ICM for refinement; the latter is faster, but could get stuck in a local minimum if used too soon. This approach is able to produce high quality quantized results. However, it is slow, e.g. taking a couple of minutes to quantize a small image (512×512 pixels) to 64 colors. Furthermore, the results contain visible noise (see Fig 1d, for example), and do not preserve colors in salient regions well. The cause of the noise is that Gaussian filtering mixes the colors of pixels within a certain radius no matter how different these colors may be, neglecting the fact that human eyes are sensitive to sudden color changes.

To address these issues, in this paper, we present a novel color quantization algorithm which again simultaneously quantizes and dithers color images (see Fig. 1e). The basis of our approach is similar to Puzicha's method [6], again minimizing a cost function considering pixel-wise differences between filtered versions of the quantized and input images.

H.-Z. Huang, K. Xu, and S.-M. Hu are with the Department of Computer Science and Technology, Tsinghua University, and Beijing Engineering Research Center for Intelligent procesing of Visual Media and Content Security, Beijing 100084, China (e-mail: huanghz08@gmail.com; xukun.1985@gmail.com; shimin@tsinghua.edu.cn). K. Xu is the corresponding author.

R. R. Martin is with the School of Computer Science & Informatics, Cardiff University, Cardiff CF24 3AA, U.K. (e-mail: ralph@cs.cf.ac.uk).

F.-Y. Huang is with the Social Network Platform Department of Tencent, Shanghai, China. (e-mail: garyhuang@tencent.com).

However, instead of using a Gaussian filter, we use edge-aware filters when defining the cost function, which avoids mixing colors with large differences, which effectively suppresses noise. Furthermore, the importance of each pixel is weighted according to its visual saliency, which helps to better preserve the fidelity of salient regions during quantization. Our approach also improves the speed with which the cost function is minimized. Instead of using DA with low temperature ICM, we use an ICM algorithm from the beginning to the end. To avoid the problem that ICM can easily converge to a local minimum, we carefully initialize ICM with a palette generated by a modified median-cut method (instead of a random palette), which turns out to be very effective. Compared to previous approaches, our method can produce high quality results with fewer visual artifacts while taking significantly less computing time.

In the context of related color quantization work [6], [7], [8], [9], our advances include:

- A novel cost function for simultaneous color quantization and dithering, which takes into account edges and saliency, leading to better visual quality.
- An efficient optimization technique for the minimization problem involved, reducing the time needed to quantize a typical image from minutes to seconds.

II. Related Work

A. Color Quantization

Color quantization has a long history, due to limitations of early computer graphics hardware. The original approaches mainly utilized clustering in color space. Representative clustering approaches have been based on median-cut [2], octrees [3], self-organizing maps [4], minmax [10], k-means [1], fuzzy c-means [11], adaptive distributing units [12], and variance-cut based on Lloyd-Max iterations [13].

To suppress the false edge artifacts caused by color quantization, dithering can be applied. Floyd-Steinberg dithering [5] is one of the most popular methods. It spreads the quantization error at each pixel to the neighboring pixels using a fixed distribution. Jarvis-Judice-Ninke dithering [14] diffuses the error both to neighbouring pixels and pixels that are one step further away, which is both slower and gives coarser dithering results.

The necessity of considering quantization and dithering as a joint problem to achieve optimal results has been noted by various authors. Orchard and Bouman [7] generate a palette using binary tree splitting and then combine modified dithering techniques with the quantization process. Scheunders [8] diffuses the quantization error to two neighboring pixels during a competitive learning process. Ozdemir and Akarun [9] combine fuzzy c-means methods with Floyd-Steinberg dithering to create a color palette, and then use this palette to perform quantization and dithering as usual. The above methods consider perform quantization and dithering as sequential steps in an iterative scheme. However, Puzicha et al. [6] was the first to *simultaneously* perform quantization and dithering by minimizing a cost-function based on a weighted Gaussian distortion measure, which aims to directly simulate human visual perception processes. In this paper, we will show that such optimization-based combined quantization and dithering can be significantly improved both in terms of speed and quality.

B. Edge-aware Filtering

One of the key ideas of our method is to replace the Gaussian filter used in [6] by an edge-aware filter, as the human visual system is sensitive to edges: our method avoids blending colors from opposite sides of an edge. Various edge aware filters exist; we use the bilateral filter [15] for its simplicity and speed of computation. For each pixel, the bilateral filter computes a weighted average of the colors of its neighbors, where the weights depend not only on spatial distances, but also on color differences. Many techniques have been proposed for accelerating the computation of the bilateral filter [16], [17]. However, although such methods are available, in our optimization process, in order to implement acceleration based on a look-up table, we need to compute and store the values of filter coefficients, instead of just filtering the image, which inevitably results in $O(Nr^2)$ time complexity, where N is the number of pixels, and r is the kernel size. Thus, brute-force computation of bilateral filtering coefficients suffices.

As well as the bilateral filter, various other edge-aware filters have also been proposed, such as anisotropic diffusion [18], median filters [19], and others [20], [21], [22], [23], [24]. Typically, they are either too inefficient or otherwise unsuitable for our cost function definition and optimization process.

C. Saliency Detection

When looking at an image, humans pay most attention to visually salient areas. Exploiting this fact, visual saliency computation has been widely used in applications such as image segmentation [25], adaptive compression [26], and image retrieval [27]. Since all color quantization methods inevitably introduce approximation, it is reasonable to give priority to salient regions, while pushing the approximation error to the less important regions. There is extensive work on saliency detection; representative works such as [28], [29]. In this paper, we use the approach in [30] for its efficiency, its ability to abstract away unnecessary details, and its generation of homogeneous saliency values across similar regions.

III. Overview

We now outline our method. We use an optimization-based process [6] to solve the combined color quantization and dithering problem, minimizing a cost function which considers pixel-wise differences between the filtered quantized image and the filtered input image. We modify the cost function in [6] to take into account edge-awareness and saliency terms (see Sec. IV). To quickly solve the minimization problem, we adopt an iterative multi-scale framework [6] in which the result for a coarser level image is used to initialize the finer level image. Instead of using deterministic annealing (DA) technique for the coarser level and only using iterative conditional modes (ICM) when the temperature is close to zero [6], we apply ICM to all levels, but still achieve a minimum which is close to that obtained by the DA+ICM method. Directly applying ICM to all levels can easily lead to convergence to a local

(a) Input image (b) k-means quantization (c) k-means + FS dithering (d) Puzicha's method (e) Our method

Fig. 1. Combined quantization and dithering to a palette of 32 colors. (a) Input image. (b) The result of k-means quantization; note the obvious false edge artifacts. (c) Using the same palette, Floyd-Steinberg dithering leads to fewer false edges. (d) The result of Puzicha's method, which preserves the original colors and suppresses false edges better than (c), but at the cost of greater noise. (e) The result of our combined quantization and dithering method, which preserves the original colors better, has fewer visible false edges, and largely suppresses noise.

minimum. We overcome this obstacle by careful choice of initial palette, obtained using a fast color quantization method: modified median-cut [31] (see Sec. V).

IV. Cost Function

This section describes our modified cost function for combined color quantization and dithering.

Problem Formulation. Given an image I with N pixels, let the color at pixel i ($1 \leq i \leq N$) be $\mathbf{c}_i$, and the desired number of colors in the palette be K. Color quantization is the process of (i) choosing a list of colors $\mathbf{p}(k)$, $1 \leq k \leq K$ making up the color palette $P = \{\mathbf{p}(k)\}$, and (ii) a pixel map $M = \{m_i, 1 \leq i \leq N\}$ indicating that pixel i uses color m_i in the palette. The goal of color quantization is to preserve the fidelity of the original image as much as possible for the given palette size K, which is achieved by minimizing a cost function.

Puzicha's Cost Function. We briefly review the cost function E used in [6]. It is defined as:

$$E = \sum_{1 \leq i \leq N} \left\| \sum_{j \in N_i} w_{ij} (\mathbf{c}_j^P - \mathbf{c}_j) \right\|^2, \tag{1}$$

where i, j denote pixels, N_i denotes the neighborhood of i, and w_{ij} is the weight of a neighboring pixel j with respect to pixel i. $\mathbf{c}_j$ and $\mathbf{c}_j^P$ denote the color (in RGB color space) of pixel j in the original image and quantized image, respectively, so $\mathbf{c}_j^P = \mathbf{p}(m_j)$. The cost function is the sum of L^2-norm differences between original and quantized color values over all pixels. However, instead of directly computing pixel-wise differences, local filtering is performed over neighborhood pixels to weight the differences. Specifically, the weight w_{ij} (or filtering kernel) is defined by a Gaussian filter:

$$w_{ij} = w_{ij}^g / \sum_{j \in N_i} w_{ij}^g, \tag{2}$$

$$w_{ij}^g = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / \sigma_s^2), \tag{3}$$

where $\mathbf{x}_i$ is the spatial location of a pixel, and σ_s is a controllable parameter. The purpose of such filtering is to simulate the spatial low-pass characteristics of the human visual system. $\sum_{j \in N_i} w_{ij} \mathbf{c}_j$ is effectively the perceived color at pixel location i. Minimizing this cost function is equivalent to performing color quantization and dithering simultaneously.

Our Cost Function. We modify the original cost function in Equation 1 in two ways. Firstly, Gaussian filtering mixes colors across either side of an edge. This neglects the fact that the human visual system is sensitive to edges, so instead we replace the Gaussian filter by a bilateral filter. Here, *edges* means any sudden color changes in general. Secondly, since humans pay more attention to salient regions, we should give higher priority to keeping the fidelity of salient regions during quantization, at the cost of sacrificing the fidelity of less salient regions. Thus, we introduce a visual importance map into the cost function.

Our modified cost function is thus:

$$E = \sum_{1 \leq i \leq N} t_i \left\| \sum_{j \in N_i} w'_{ij} (\mathbf{c}_j^P - \mathbf{c}_j) \right\|^2, \tag{4}$$

where the weight w'_{ij} is now defined by a normalized bilateral filtering kernel:

$$w'_{ij} = w_{ij}^b / \sum_{j \in N_i} w_{ij}^b, \tag{5}$$

$$w_{ij}^b = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / \sigma_s^2) \exp(-\|\mathbf{c}_i - \mathbf{c}_j\|^2 / \sigma_r^2), \tag{6}$$

and σ_s and σ_r are two parameters to control the relative contribution of spatial and color components. The importance value t_i is defined according to visual saliency values:

$$t_i = \lambda + (1 - \lambda) s_i, \tag{7}$$

where we set $\lambda = 0.1$ in order to avoid giving pixels very small or zero importance values, and s_i is the visual saliency of pixel i computed using the saliency model in [30] for its efficiency and generation of homogeneous saliency values across similar regions. Other saliency models could also be used, and, for example, a face detector could additionally be used to to assign larger importance values to human face regions. We refer to the value of this cost function as the *edge-aware spatial quantization error*, ESQE for short.

V. Optimization

Recall that the quantized color $\mathbf{c}_j^P$ in the cost function (Equation 4) is computed from the color palette P and the pixel map M. We rewrite the cost function by replacing $\mathbf{c}_j^P$ by $\mathbf{p}(m_j)$:

$$E = \sum_{1 \leq i \leq N} t_i \left\| \sum_{j \in N_i} w'_{ij} (\mathbf{p}(m_j) - \mathbf{c}_j) \right\|^2. \tag{8}$$

The unknowns here are the color palette $P = \{\mathbf{p}_k\}$ and the pixel map $M = \{m_j\}$. This is a nonlinear, mixed combinatorial minimization target.

We solve it in an iterative manner. In each iteration, we first fix the palette P and solve for the pixel map M, then we fix the pixel map M and solve for the palette P. Finding the pixel map is a combinatorial optimization problem while finding the color palette requires solution of a linear system. We stop iterating when the results converge. We also utilize a multi-scale scheme for efficiency. Details of our algorithm are explained below; pseudocode is given in Algorithm 1.

Finding the pixel map. During this stage, we fix the palette P and optimize the pixel map M. We employ the iterative conditional mode (ICM) algorithm [32] to do so. Specifically, we enumerate over all pixels in a random order. At each time, we update the color index m_i of a specific pixel i while keeping the color indices of all other pixels unchanged. We change the color index m_i to the new value m_i' in the range $1 \leq m_i' \leq K$ which minimizes the cost function (Equation 8). A naive approach takes time $O(KN)$ to obtain the new color indices (K evaluations for N pixels), and is prohibitively expensive. Two schemes are used for acceleration. Firstly, to compute the cost function, instead of fully evaluating it using Equation 8, taking time $O(N)$, we employ the *local updating scheme* proposed in [6] to incrementally compute it using a look-up table. When we change the color index m_i of a single pixel i, only the neighborhood of that pixel is affected. Hence, we can record the original value of the cost function before updating, and only re-evaluate a small part of the cost function (i.e. for pixels in the neighborhood), to incrementally update its value. Secondly, instead of testing all K colors in the color palette, we use a *greedy scheme* which only tests a small number of colors in the palette which are similar to the current assigned pixel color. The most similar colors (in the palette) for each color in the palette are precomputed and stored before solving for the pixel map. In our implementation, we set the number of similar colors to $n = 10$ for a palette size 32, which achieves a good trade-off between efficiency and fidelity. The effect of varying n is shown in Section VI. After considering all pixels, we check the number of pixels that have changed. If the proportion of changed pixels is lower than a predefined threshold t_m (in experiments, we set $t_m = 0.001$), we cease updating pixels.

Algorithm 1 Calculate palette and pixel map

Input: original image I
Output: pixel map M, color palette P

Initialize P using modified median-cut.
Initialize M randomly.
$l \Leftarrow 1$
$I_1 \Leftarrow I$
while $l < l_{\max}$ **do**
  $I_{l+1} \Leftarrow \text{subsample}(I_l)$
  $l \Leftarrow l + 1$
end while
$l \Leftarrow l_{\max}$
repeat
  repeat
    repeat
      $N_l \Leftarrow$ number of pixels in I_l
      $\pi \Leftarrow$ a random permutation of $1, \ldots, N^l$
      for $i = 1, \ldots, N^l$ **do**
        Update $m_{\pi(i)} \in M$ using ICM
      end for
    until M converges
    Update P by solving a linear system
  until P converges
  if $l > 1$ **then**
    $M \Leftarrow \text{upsample}(M)$
  end if
  $l \Leftarrow l - 1$
until $l = 0$

Finding the color palette. In this stage, we fix the pixel map M and optimize the color palette P. As in [6], this is done by directly solving a small linear system derived from:

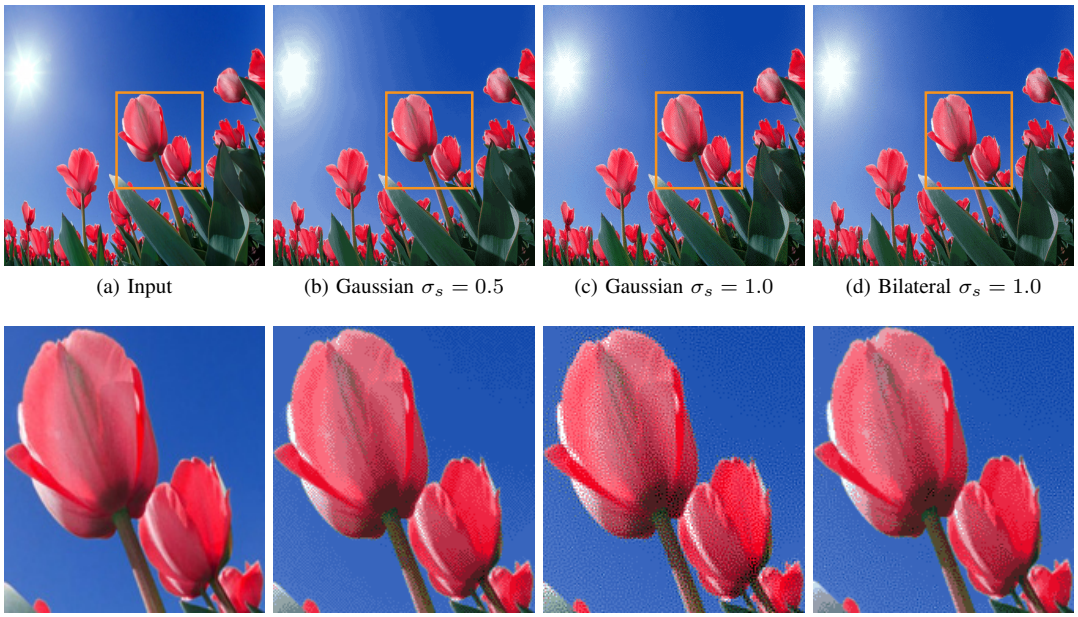
$$\frac{\partial E}{\partial P_{ks}} = 0, \tag{9}$$

where P_{ks} denotes the k-th color of the palette P in RGB channel s.

Termination. At the end of each iteration, we check how many colors have changed in the color palette since the last iteration. A color in the palette is considered to have changed if the difference is larger than 1 (color values are in the range 0–255). If the proportion of new colors in the palette is smaller than a predefined threshold t_p (in experiments, we set $t_p = 0.1$), we regard the process as having converged and terminate the iteration.

Multi-scale Solution. Inspired by [6], we also use a multi-scale framework to accelerate the optimization process. We build an image pyramid from the input image by iterative subsampling by a factor of two. Following Puzicha's setting, we use a 5 level multi-scale framework ($l_{\max} = 5$). We start the optimization at the coarsest level. The pixel map resulting from one level is upsampled and used to initialize the next level. Specifically, the pixel map value at (x, y) of the finer scale is copied from $(floor[x/2], floor[y/2])$ of the coarser level, as in [6]. The color palette is copied directly to the next level.

Initialization. The convergence and efficiency of the above iterative optimization process depend greatly on the values used to initialize the color palette. A straight-forward approach is to use a random palette, as done in [6]. However, in experiments we find that this can cause the optimization process to become stuck in a local minimum. To address this issue, we employ a modified median-cut (MMC) algorithm adapted from [31] to generate the initial palette, since, to our knowledge, MMC acquires the lowest MSE among traditional clustering-based quantization methods. Specifically, first, each pixel is assigned a visual importance weight using its saliency value [30]. Next, we place a bounding-box in RGB color space surrounding all colors in the image and iteratively split a box at the median value along its longest axis, until the number of boxes is equal to the size of the palette. Each time splitting occurs, we choose the box with the largest product of the



(a) Input (b) Gaussian $\sigma_s = 0.5$ (c) Gaussian $\sigma_s = 1.0$ (d) Bilateral $\sigma_s = 1.0$

(e) Input (f) Gaussian $\sigma_s = 0.5$ (g) Gaussian $\sigma_s = 1.0$ (h) Bilateral $\sigma_s = 1.0$

Fig. 2. Comparison between Gaussian and bilateral kernels. Top: images, bottom: zoomed detail. (a) Input image. While (b) contains false edges, (c) contains too much noise; (d) shows that our method avoids false edges and suppress noise.

sum of the visual importance weights in it and variance of colors in it. After the splitting stage, the average color of each box is used as an initial color in the palette. After median-cut, the palette is further adjusted using k-means, which iteratively recalculates the nearest palette color for each color under the L^2-norm and updates the palette with the new cluster centers. Hybridizing median-cut and k-means can utilize the speed of median-cut and the flexibility of cluster boundaries of k-means. After a round of median-cut and k-means, the visual importance weights of each pixels are increased by a factor proportional to their distance from their nearest palette color, and then do a second round of median-cut and k-means. This is iterated until the MSE between the input image and the current result image fails to improve for several rounds. For more details, please see [31]. Using the palette resulting from this modified median-cut algorithm, our optimization algorithm is able to produce results with comparable quality to the DA algorithm [6], but is much faster.

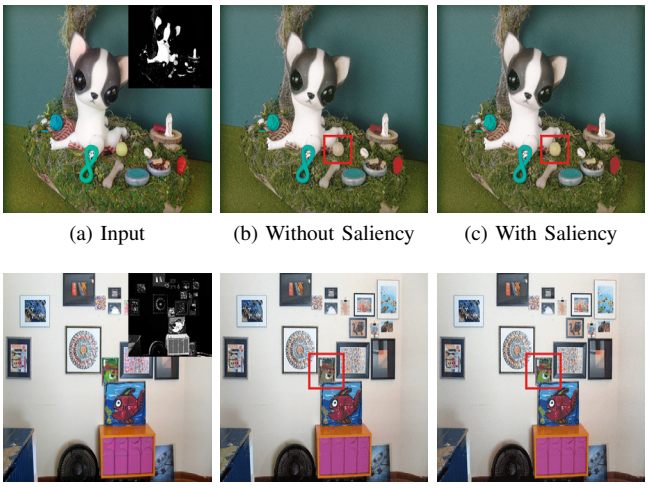
Differences from Puzicha's method. Our optimization method differs from Puzicha's method [6] in several ways. Firstly, when solving for the pixel map, they use deterministic annealing. Since DA is slow, for speed, they switch to using the iterative conditional mode algorithm once the annealing temperature is close to zero. In contrast, we use the iterative conditional mode algorithm throughout, allowing us to find the pixel map much more quickly. As mentioned above, we also use a greedy scheme for further acceleration. Secondly, they initialize the color palette randomly, while we use a more sophisticated method which generates better initialization, which is necessary to be able to use ICM optimization throughout. While their method typically needs about 20 minutes to generate a quantized result with 256 colors for a small image (512×512 pixels), our method can do so under 12 seconds.

VI. Experiments

We have implemented our method in C++ using the OpenCV library on a PC with an Intel 3.4GHz Core i7-3770 CPU. Source code is available at our project page. By default, the parameters used in our experiments were $\sigma_s = 1.0$ (spatial distance is measured in pixels), $\sigma_r = 2.0$ (color range for each channel of RGB space is $[0, 255]$), kernel size 3×3, palette size 32, $n = 10$. The 100 test images used as inputs in our experiments were downloaded from [33], Wikipedia and Flickr. They were all resized or cropped to 512×512 to ease comparison.

Evaluation of Our Cost Function. We now evaluate the effectiveness of our cost function, which replaces the original Gaussian kernel based cost function in [6], to one based on a bilateral filtering kernel, and also includes a saliency term. In Fig. 2, we compare the quantized results generated using a bilateral filtering kernel to those generated using Gaussian kernels with different σ_s. We can see that when using Gaussian kernels, setting $\sigma_s = 0.5$ is unable to remove false edge artifacts in the smoothly changing area. Increasing σ_s to 1.0 removes false edges but introduces significant noise. In contrast, the results generated by using bilateral filtering kernels achieve a good balance between suppressing false edges and suppressing noise. In Fig. 3, we compare quantized results generated with and without the saliency term. Using the saliency term preserves better fidelity in salient areas.

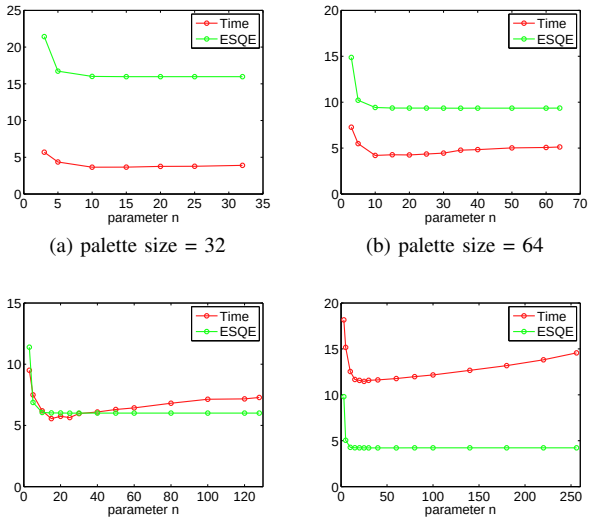
Parameter settings. In solving for pixel maps, we employ a greedy scheme which only considers the n most similar colors in the palette. We generated results using different values of n and record the edge-aware spatial quantization error (ESQE) after convergence. The variation of ESQE with n is shown in Fig. 4. For a palette size of 256, $n = 25$ achieves a good trade-off between speed and quality. $n = 10, 10, 15$ are the



(a) Input (b) Without Saliency (c) With Saliency

(d) Input (e) Without Saliency (f) With Saliency

Fig. 3. The effect of saliency weights. (a), (d): input images with saliency maps shown at top-right. With the guidance of the saliency map, (c) preserves the yellow color of the ball (marked by a red rectangle) better than (b). Similarly, (f) preserves the green color of the picture better than (e).



(a) palette size = 32 (b) palette size = 64

(c) palette size = 128 (d) palette size = 256

Fig. 4. The effect of parameter n. For palette size of 256, the fastest speed without sacrificing quality is achieved at about $n = 25$. Similarly, $n = 10, 10, 15$ are best values for palette sizes of 32, 64 and 128, respectively.

best values for palette sizes of 32, 64 and 128, respectively.

Secondly, we tested the effect of different kernel sizes. In our experiments, by changing the kernel size from 5×5 to 3×3, the average time for processing the 100 test images significantly decreased from 8.9 s to 3.7 s without visible quality loss. So, 3×3 kernels are used by default. The same kernel size was used for Puzicha's method[6] while performing comparisons.

Comparison with other methods. We next compared our method (all levels ICM optimization with palette initialized using modified median-cut, or MMC+AICM for short) with several existing methods, including quantization using k-means (KM) [1], quantization using modified median-cut (MMC) [31], quantization using adaptive distributing

TABLE I
METHODS COMPARISON, AVERAGE OVER 100 TEST IMAGES

Method	Time	MSE	SSIM	SQE	ESQE
KM	0.18 s	78.5	0.88	44.3	26.7
KM+FS	0.21 s	107.6	0.77	27.3	27.2
MMC	1.7 s	62.0	0.89	31.4	21.3
MMC+FS	1.8 s	87.7	0.81	19.3	23.5
ADU	0.03 s	63.5	0.89	31.5	23.4
ADU+FS	0.07 s	89.3	0.83	19.0	25.6
VCL	0.02 s	70.5	0.89	37.2	24.6
VCL+FS	0.06 s	101.4	0.78	22.3	26.8
DA+ICM	46.0 s	335.6	0.59	6.1	85.7
MMC+AICM(GAUS)	5.8 s	339.8	0.58	6.1	77.0
RAND+AICM	10.3 s	164.9	0.70	31.0	21.8
MMC+AICM	3.6 s	99.0	0.77	21.3	16.1

units (ADU) [12], quantization using variance-cut based on Lloyd-Max iterations (VCL) [13], quantization using k-means followed by Floyd Steinberg dithering [5] (KM+FS), quantization using modified median-cut followed by Floyd Steinberg dithering (MMC+FS), quantization using ADU followed by Floyd Steinberg dithering (ADU+FS), quantization using VCL followed by Floyd Steinberg dithering (VCL+FS) and Puzicha's [6] deterministic annealing with ICM at low temperatures with Gaussian filtering weights (DA+ICM). To demonstrate that our optimization method can achieve comparable quality to Puzicha's DA+ICM method, we also generated results using our MMC+AICM optimization method with Puzicha's cost function, which are denoted by MMC+AICM(GAUS). To demonstrate the importance of a good palette initialization instead of random initialization, we also give results of our method with random initialization (RAND+AICM).

For each test image, we generated a quantized result using each of the above methods. In the absence of a perfect psychophysically defined error measure [34], for each quantized image, we report quantitive results under various perceptual metrics including the most popular mean squared error (MSE), structural similarity (SSIM) [35], Puzicha's spatial quantization error (SQE) [6] and the edge-aware spatial quantization error (ESQE) derived from our cost function. In the comparison, we set the dithering level in the Floyd Steinberg method to 1.0, the number of iterations in k-means to 10, and the number of iterations in MMC to 30. Here k-means algorithm is randomly initialized. Specifically, the initial center of each cluster is randomly chosen from one of the existing colors. For each method, the average time to generate a result, and average MSE, SQE and ESQE values over all 100 test images are recorded. For randomized methods, we ran the test 10 times for each input image and recorded the mean values of these qunatities. Results are reported in Table I. Table II gives detailed statistics for all the images which appear in this paper. We can see that the MSE values become larger and SSIM values become smaller after FS dithering for KM, MMC, ADU and VCL. Since KM, MMC, ADU and VCL without FS dithering cause obvious false edges, it suggests that MSE and SSIM are imperfect assessment methods for the color quantization problem. Among the traditional clustering based algorithms (KM, MMC, ADU and VCL), MMC gives

(a) Input (b) KM (c) ADU (d) VCL

(e) MMC (f) KM+FS (g) ADU+FS (h) VCL+FS

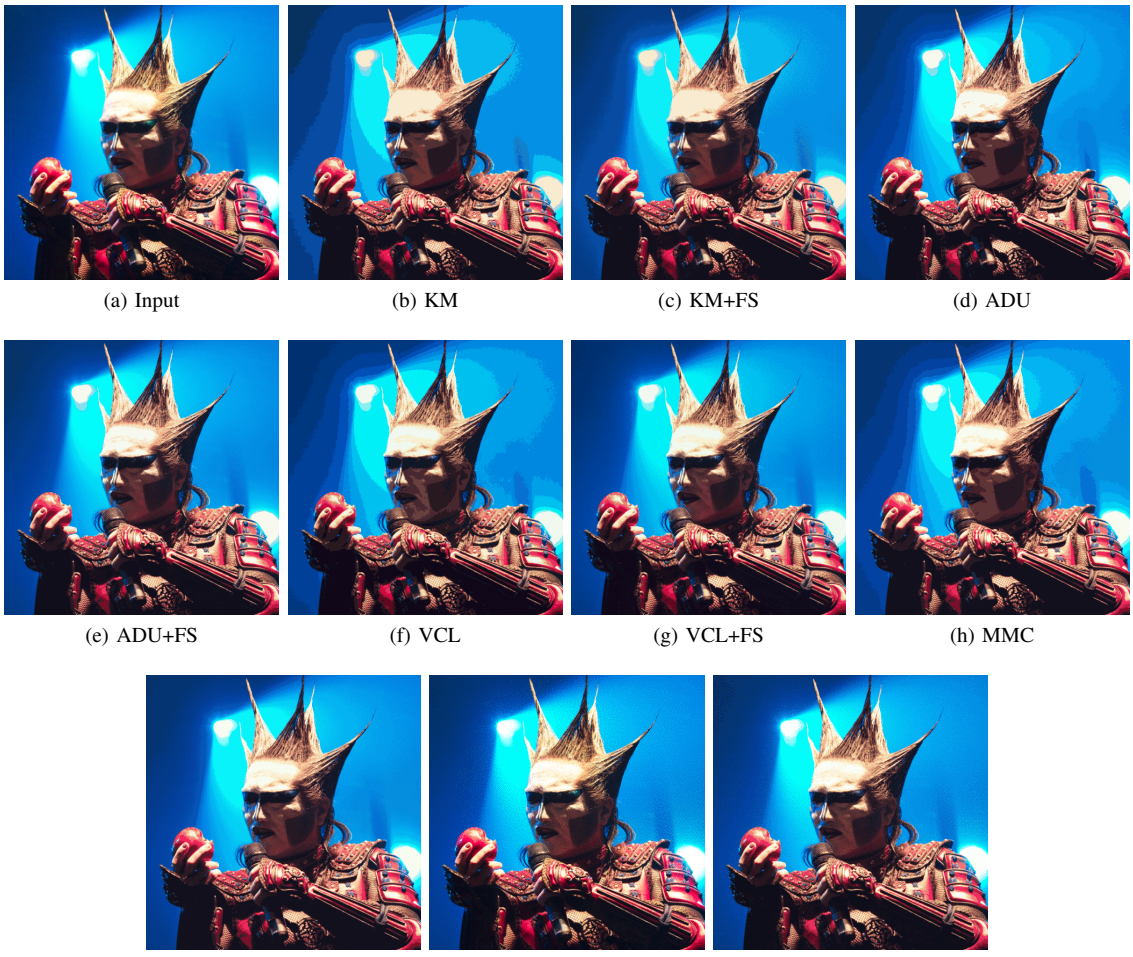
(i) MMC+FS (j) DA+ICM (k) MMC+AICM(ours)

Fig. 5. Comparison of different optimization methods. Using Floyd-Steinberg dithering, (f-i) hide some of the false edges seen in (b-e), but there are still some obvious false edges on the car. Both DA+ICM (j) and our method (k) achieve better results with fewer false edges. However, our method (k) introduces less noise than Puzicha's method (j).

TABLE II
METHODS COMPARISON FOR EACH IMAGE

Method	Fruit (Fig 7 (a))					Wall (Fig 7 (b))					Butterfly (Fig 7 (c))					Parrot (Fig 7 (d))					Church (Fig 7 (e))				
	Time	MSE	SSIM	SQE	ESQE	Time	MSE	SSIM	SQE	ESQE	Time	MSE	SSIM	SQE	ESQE	Time	MSE	SSIM	SQE	ESQE	Time	MSE	SSIM	SQE	ESQE
KM	0.24 s	133.8	0.78	80.5	23.8	0.28 s	72.5	0.90	29.9	15.4	0.13 s	78.7	0.94	60.9	16.7	0.19 s	105.9	0.83	70.7	38.4	0.20 s	80.6	0.92	43.5	19.0
KM+FS	0.26 s	195.5	0.63	46.1	24.0	0.29 s	96.33	0.86	20.9	17.0	0.17 s	161.3	0.70	44.1	16.7	0.21 s	181.4	0.65	77.13	46.3	0.22 s	119.3	0.78	25.4	21.1
MMC	1.9 s	126.1	0.78	75.7	21.8	2.2 s	68.6	0.91	28.3	15.3	0.84 s	49.7	0.94	32.8	13.5	1.3 s	77.4	0.83	42.6	30.5	1.6 s	59.2	0.92	23.7	14.0
MMC+FS	1.9 s	182.9	0.64	44.0	22.2	2.3 s	91.1	0.86	19.2	17.8	0.9 s	86.6	0.74	16.3	13.8	1.3 s	105.9	0.70	27.9	33.0	1.6 s	84.6	0.83	12.0	16.6
ADU	0.03 s	126.7	0.79	74.4	23.3	0.03 s	69.6	0.91	28.9	13.6	0.03 s	43.9	0.93	21.2	17.8	0.03 s	80.8	0.84	39.3	29.2	0.03 s	62.6	0.93	24.7	16.4
ADU+FS	0.05 s	184.6	0.65	41.6	23.7	0.07 s	91.9	0.87	19.9	15.8	0.06 s	61.5	0.86	13.3	21.1	0.06 s	106.3	0.79	24.2	31.5	0.07 s	91.0	0.88	13.8	21.6
VCL	0.03 s	140.3	0.78	86.7	24.7	0.03 s	75.6	0.91	36.5	15.9	0.02 s	50.3	0.93	29.9	16.5	0.02 s	78.3	0.85	40.9	31.5	0.02 s	68.7	0.91	30.5	18.8
VCL+FS	0.05 s	205.1	0.63	49.1	24.8	0.07 s	100.3	0.86	26.2	18.7	0.05 s	73.4	0.80	18.1	18.7	0.05 s	111.6	0.72	22.6	34.5	0.06 s	98.5	0.84	17.1	22.5
DA+ICM	41.5 s	618.2	0.38	11.5	63.3	37.6 s	431.9	0.60	7.1	69.5	55.2 s	305.3	0.49	6.1	54.6	47.3 s	406.5	0.43	8.3	92.4	37.9 s	359.9	0.68	6.5	95.9
MMC+AICM(GAUS)	5.5 s	623.4	0.37	11.2	55.4	5.5 s	401.7	0.60	6.5	68.7	6.9 s	305.8	0.53	5.9	57.6	4.9 s	307.5	0.53	6.1	78.0	4.3 s	370.2	0.61	5.9	76.1
RAND+AICM	5.8 s	237.3	0.59	50.6	16.6	7.0 s	173.4	0.80	63.8	20.7	14.0 s	537.2	0.43	39.4	15.3	5.4 s	190.7	0.62	48.7	27.0	8.8 s	118.1	0.75	25.5	15.7
MMC+AICM (ours)	3.3 s	214.0	0.59	44.3	14.7	3.1 s	89.8	0.87	31.3	12.4	3.2 s	115.8	0.67	16.8	9.6	2.9 s	112.0	0.66	23.1	18.8	3.1 s	83.8	0.79	18.1	12.2
Method	Harbor (Fig 1)					Puppy (Fig 3 top)					Room (Fig 3 bottom)					Flower (Fig 2)					Car (Fig 5)				
	Time	MSE	SSIM	SQE	ESQE	Time	MSE	SSIM	SQE	ESQE	Time	MSE	SSIM	SQE	ESQE	Time	MSE	SSIM	SQE	ESQE	Time	MSE	SSIM	SQE	ESQE
KM	0.19 s	87.7	0.91	56.4	18.0	0.19 s	73.3	0.86	43.6	23.4	0.17 s	91.9	0.88	56.0	25.8	0.15 s	98.4	0.89	67.9	27.9	0.18 s	98.6	0.85	59.9	31.4
KM+FS	0.21 s	127.7	0.68	26.4	13.4	0.23 s	105.8	0.74	26.1	21.2	0.20 s	127.9	0.68	23.8	27.3	0.18 s	102.7	0.72	23.2	22.6	0.20 s	146.1	0.71	38.8	32.9
MMC	1.9 s	65.4	0.91	35.6	11.6	1.8 s	52.0	0.90	27.1	11.3	1.2 s	73.5	0.89	40.0	26.2	1.4 s	61.1	0.89	33.0	21.5	1.4 s	69.5	0.87	35.6	24.1
MMC+FS	1.9 s	97.8	0.76	19.0	11.4	1.8 s	77.0	0.80	18.2	12.0	1.3 s	109.8	0.74	21.0	29.5	1.4 s	86.6	0.74	17.0	21.8	1.4 s	100.7	0.76	19.7	27.3
ADU	0.03 s	67.3	0.91	35.0	11.8	0.03 s	54.4	0.90	28.8	11.5	0.03 s	76.4	0.89	40.5	26.8	0.03 s	79.4	0.88	42.8	36.2	0.03 s	73.6	0.88	33.1	28.9
ADU+FS	0.05 s	98.4	0.80	20.1	12.3	0.07 s	76.4	0.83	19.6	12.0	0.06 s	111.9	0.77	22.2	29.6	0.06 s	112.3	0.78	23.7	39.4	0.07 s	106.4	0.79	18.4	32.6
VCL	0.02 s	73.5	0.90	41.4	13.9	0.03 s	63.7	0.88	34.6	13.9	0.02 s	84.5	0.89	51.8	29.4	0.02 s	70.8	0.88	40.5	25.8	0.02 s	78.8	0.87	39.1	29.5
VCL+FS	0.04 s	112.3	0.72	21.4	12.8	0.07 s	92.6	0.78	21.7	14.2	0.05 s	118.5	0.77	34.3	32.2	0.05 s	101.4	0.70	20.4	25.5	0.06 s	117.9	0.75	19.8	32.5
DA+ICM	41.6 s	376.9	0.53	6.8	39.0	34.1 s	277.1	0.55	4.0	37.3	47.6 s	390.6	0.48	8.4	80.9	27.6 s	244.6	0.54	6.2	64.1	30.8 s	337.8	0.57	6.3	96.7
MMC+AICM(GAUS)	5.9 s	342.9	0.52	6.2	34.7	5.7 s	265.4	0.52	6.4	33.0	6.2 s	417.9	0.46	7.4	76.0	5.1 s	289.0	0.48	6.0	61.6	3.9 s	365.6	0.55	6.9	89.1
RAND+AICM	7.2 s	166.1	0.61	26.3	10.1	23.0 s	113.1	0.69	23.1	9.0	10.1 s	213.9	0.56	29.7	20.0	25.3 s	271.6	0.50	17.0	15.9	3.9 s	210.3	0.65	21.3	19.1
MMC+AICM (ours)	3.4 s	111.0	0.67	20.1	7.7	3.1 s	78.2	0.75	20.4	7.4	5.9 s	179.7	0.57	22.7	17.7	4.5 s	126.2	0.61	16.7	15.1	2.5 s	126.3	0.69	19.8	18.1

results with the lowest MSE, which is why we use MMC to initialize AICM. If users want a faster initialization with higher MSE, ADU may also be a good choice. Comparing DA+ICM with MMC+AICM(GAUS), we can see that our optimization



(a) Input (b) KM (c) KM+FS (d) ADU

(e) ADU+FS (f) VCL (g) VCL+FS (h) MMC

(i) MMC+FS (j) DA+ICM (k) MMC+AICM(ours)

Fig. 6. Another comparison of different optimization methods. In this example, it is clearer that generally our method (k) hides false edges better than (f-i), especially the false edges around those lights. (k) also introduces less noise than Puzicha's method (j).

method can achieve similar quality to Puzicha's method but takes less time. Comparing RAND+AICM with MMC+AICM, we can see that the initial palette from MMC improves AICM both in speed and quality.

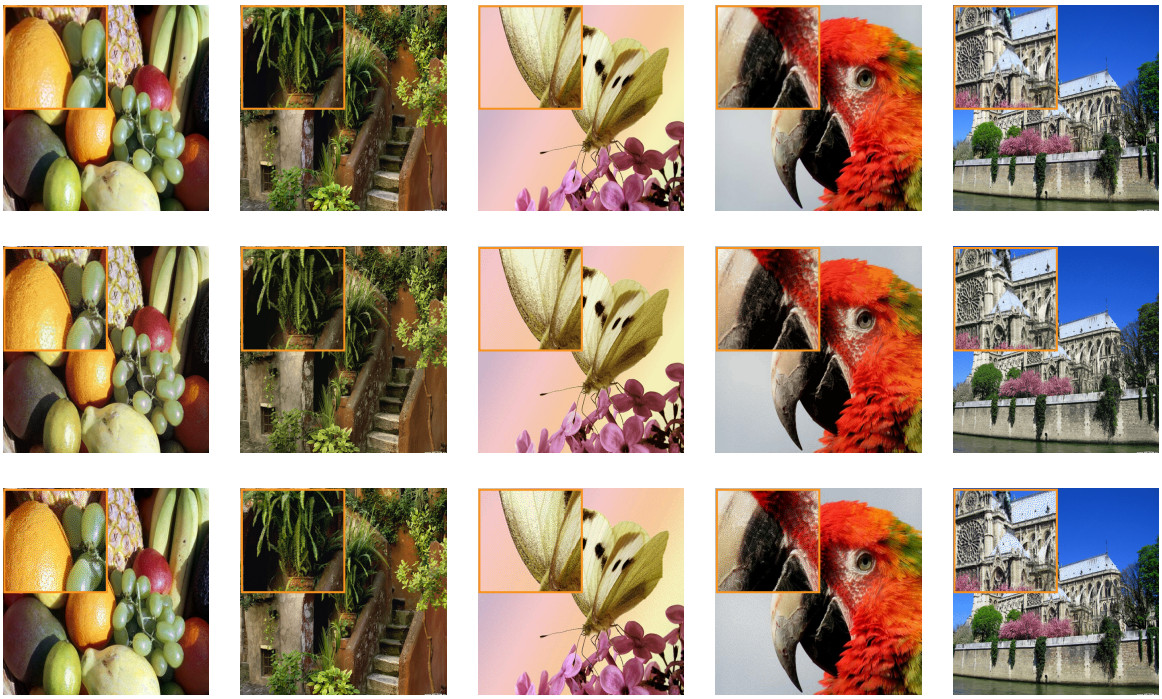
Fig. 5 and Fig. 6 shows output quantized images using different methods. We can see that KM, MMC, ADU and VCL produce visible false edge artifacts, which FS dithering can hide to some degree, but not always: see the false edges on the red car (Fig. 5) and around the lights (Fig. 6). In comparison, generally smooth results are produced by DA+ICM and MMC+AICM (our method) for all parts of the image, but introduce some noise. Generally speaking, false edges are more visible than noise when we look at an image from some distance. Moreover, MMC+AICM introduces less noise than DA+ICM. Thus we believe our method (MMC+AICM) to be more preferable. Further results are given in the supplementary material.

To further explore the efficiency of our method, we compare the timings of MMC+FS, DA+ICM and MMC+AICM (our method) with different sizes of color palette. As shown in Table III, with a palette size of 256, our MMC+AICM method takes less than 12 seconds, while DA+ICM takes more than 20 minutes. Thus, our method produces quantized results with better quality while taking significantly less time than DA+ICM. While MMC+FS takes less time but introduces more false edges, our method takes a little longer, but introduces some noise. However, false edges are more visible than noise. Given the small performance difference, we believe our method is more preferable.

TABLE III
THE SPEED OF DIFFERENT SIZE OF COLOR PALETTE

Colors	MMC+FS	DA+ICM	MMC+AICM
32	1.8 s	46 s	3.6 s
64	2.3 s	178 s	4.2 s
128	2.9 s	498 s	5.7 s
256	3.3 s	1225 s	11.6 s

Sensitivity to initial random pixel map. Recall that our pixel map is initialized randomly (see Algorithm 1). However, the results of our method are insensitive to the initial random pixel map. To demonstrate it, we have run our algorithm 10 times on 5 images. Each time, the pixel map is initialized with different random values. The mean and standard deviation (shown as $\pm x$) of the timings and errors are shown in Table IV. From the results, we can easily find that our method is insensitive to the initial random pixel map.



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

Fig. 7. Visual comparison between our method and Puzicha's method. Top: input images. Middle: results of our method. Bottom: results using Puzicha's method [6]. Zoom-ins of the centers of each image are shown at the top-left corner. The size of the palette is 32.

TABLE IV
MEAN AND STANDARD DEVIATION OVER 10 TIMES OF TRIALS

Image	Time	MSE	SSIM	SQE	ESQE
Fruit (Fig 7 (a))	3.3 ± 0.24 s	214.0 ± 1.29	0.59 ± 0.0015	44.3 ± 0.21	14.7 ± 0.025
Wall (Fig 7 (b))	3.1 ± 0.44 s	89.8 ± 0.50	0.87 ± 0.0019	31.3 ± 0.29	12.4 ± 0.053
Butterfly (Fig 7 (c))	3.2 ± 0.11 s	115.8 ± 0.23	0.67 ± 0.0003	16.8 ± 0.02	9.6 ± 0.003
Parrot (Fig 7 (d))	2.9 ± 0.07 s	112.0 ± 0.57	0.66 ± 0.0021	23.1 ± 0.06	18.8 ± 0.022
Church (Fig 7 (e)	3.1 ± 0.13 s	83.8 ± 0.45	0.79 ± 0.0017	18.1 ± 0.10	12.2 ± 0.029

User Study. We also carried out a subjective user study to compare our method to Puzicha's method [6]. We invited 8 males and 3 females, aged from 21 to 28, to participate in our study. 25 test images from [33] were used. For each test image, three images, including the input image, the quantized result of our method, and that produced by Puzicha's method, were shown to each participant; the two result images were shown in a random order. Each participant was asked to mark which image better preserved the fidelity of the input image (or to state that they were equally good). For 88% of the images, the participants preferred our results. Five examples are shown in Fig. 7. Generally speaking, our method preserves original colors well, successfully hides false edges in smoothly changing areas, and suppresses noise.

VII. CONCLUSION

In this paper, we have presented a novel combined quantization and dithering method. It optimizes a novel cost function based on a bilateral filtering kernel to suppress noise, and extended to include a saliency term which is able to better preserve the quality of more salient regions during quantization. For efficiency of optimization, we use an all-level iterative conditional mode algorithm with an effective initial palette generated by a modified median-cut method. Compared to previous methods, our method can produce high quality results with fewer visual artifacts while taking significantly less time, enabling practical applications for higher image resolutions and larger palette sizes.

ACKNOWLEDGMENT

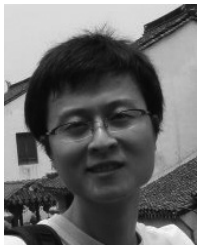
We thank the reviewers for their precious comments. We also thank M. E. Celebi, Kornel Lesiski et al. for making their code available for comparison. Many input images we use here come from other researchers, we thank them for making their data available. This work was supported by the National High Technology Research and Development Program of China (Project Number 2013AA013903), the Natural Science Foundation of China (Project Number 61133008, Project Number 61170153), Research Grant of Beijing Higher Institution Engineering Research Center, Tsinghua University Initiative Scientific Research Program, and EPSRC Travel Grant.

REFERENCES

[1] M. E. Celebi, "Improving the performance of k-means for color quantization," *Image and Vision Computing*, vol. 29, no. 4, pp. 260–271, 2011.
[2] P. Heckbert, "Color image quantization for frame buffer display," in *Proceedings of SIGGRAPH 1982*. ACM, 1982, pp. 297–307.
[3] M. Gervautz and W. Purgathofer, "A simple method for color quantization: Octree quantization," in *New Trends in Computer Graphics*. Springer, 1988, pp. 219–231.
[4] A. H. Dekker, "Kohonen neural networks for optimal colour quantization," *Network: Computation in Neural Systems*, vol. 5, no. 3, pp. 351–367, 1994.
[5] R. Floyd and L. Steinber, "An adaptive algorithm for spatial gray-scale," in *Proceedings of Society for Information Display*, vol. 17, 1976, pp. 75–77.

[6] J. Puzicha, M. Held, J. Ketterer, J. M. Buhmann, and D. W. Fellner, "On spatial quantization of color images," *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 666–682, 2000.
[7] M. T. Orchard and C. A. Bouman, "Color quantization of images," *IEEE Transactions on Signal Processing*, vol. 39, no. 12, pp. 2677–2690, 1991.
[8] P. Scheunders, "Joint quantisation and error diffusion of colour images using competitive learning," in *Proceedings of Vision, Image and Signal Processing*, vol. 145, no. 2. IET, 1998, pp. 137–140.
[9] D. Ozdemir and L. Akarun, "Fuzzy algorithms for combined quantization and dithering," *IEEE Transactions on Image Processing*, vol. 10, no. 6, pp. 923–931, 2001.
[10] Z. Xiang, "Color image quantization by minimizing the maximum intercluster distance," *ACM Transactions on Graphics*, vol. 16, no. 3, pp. 260–276, 1997.
[11] D. Özdemir and L. Akarun, "A fuzzy algorithm for color quantization of images," *Pattern Recognition*, vol. 35, no. 8, pp. 1785–1791, 2002.
[12] M. E. Celebi, S. Hwang, and Q. Wen, "Color quantization using the adaptive distributing units algorithm," *Imaging Science Journal*, vol. 60, no. 2, pp. 80–91, 2014.
[13] M. E. Celebi, Q. Wen, and S. Hwang, "An effective real-time color quantization method based on divisive hierarchical clustering," *Journal of Real-Time Image Processing*, vol. 10, no. 2, pp. 329–344, 2015.
[14] J. F. Jarvis, C. N. Judice, and W. Ninke, "A survey of techniques for the display of continuous tone pictures on bilevel displays," *Computer Graphics and Image Processing*, vol. 5, no. 1, pp. 13–40, 1976.
[15] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of IEEE ICCV*. IEEE, 1998, pp. 839–846.
[16] A. Adams, J. Baek, and M. A. Davis, "Fast high-dimensional filtering using the permutohedral lattice," vol. 29, no. 2, pp. 753–762, 2010.
[17] Q. Yang, "Recursive bilateral filtering," in *Proceedings of ECCV*. Springer, 2012, pp. 399–413.
[18] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on PAMI*, vol. 12, no. 7, pp. 629–639, 1990.
[19] S. Perreault and P. Hébert, "Median filtering in constant time," *IEEE Transactions on Image Processing*, vol. 16, no. 9, pp. 2389–2394, 2007.
[20] E. S. Gastal and M. M. Oliveira, "Domain transform for edge-aware image and video processing," *ACM Transactions on Graphics*, vol. 30, no. 4, p. 69, 2011.
[21] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE Transactions on PAMI*, vol. 35, no. 6, pp. 1397–1409, 2013.
[22] J. Xu, X. Feng, Y. Hao, and Y. Han, "Adaptive variational models for image decomposition," *Science China Information Sciences*, vol. 57, no. 2, pp. 1–8, 2014.
[23] Y. Zang, H. Huang, and L. Zhang, "Efficient structure-aware image smoothingby local extrema on space-filling curve," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 9, pp. 1253–1265, 2014.
[24] P. Shao, S. Ding, L. Ma, Y. Wu, and Y. Wu, "Edge-preserving image decomposition via joint weighted least squares," *Computational Visual Media*, vol. 1, no. 1, pp. 37–47, 2015.
[25] M. Donoser, M. Urschler, M. Hirzer, and H. Bischof, "Saliency driven total variation segmentation," in *Proceedings of IEEE ICCV*. IEEE, 2009, pp. 817–824.
[26] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The jpeg2000 still image coding system: an overview," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 4, pp. 1103–1127, 2000.
[27] T. Chen, P. Tan, L.-Q. Ma, M.-M. Cheng, A. Shamir, and S.-M. Hu, "Poseshop: human image database construction and personalized content synthesis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 5, pp. 824–837, 2013.
[28] F. Perazzi, P. Krähenbühl, Y. Pritch, and A. Hornung, "Saliency filters: Contrast based filtering for salient region detection," in *Proceedings of IEEE CVPR*. IEEE, 2012, pp. 733–740.
[29] M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S. Hu, "Global contrast based salient region detection," *IEEE Transactions on PAMI*, vol. 37, no. 3, pp. 569–582, 2015.
[30] M.-M. Cheng, J. Warrell, W.-Y. Lin, S. Zheng, V. Vineet, and N. Crook, "Efficient salient region detection with soft image abstraction," in *Proceedings of IEEE ICCV*. IEEE, 2013, pp. 1529–1536.
[31] K. Lesiski. (2015) pngquant. [Online]. Available: https://pngquant.org/
[32] J. Besag, "On the statistical analysis of dirty pictures," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 259–302, 1986.
[33] M. Hassan. (2012) Colorquantizationdata. [Online]. Available: http://dcis.uohyd.ernet.in/ hassan/
[34] Y. Ding, Y. Zhang, X. Wang, X. Yan, and A. S. Krylov, "Perceptual image quality assessment metric using mutual information of gabor features," *Science China Information Sciences*, vol. 57, no. 3, pp. 1–9, 2014.
[35] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

Hao-Zhi Huang is currently a Ph. D. student in the Department of Computer Science and Technology, Tsinghua University. He received a Bachelor's degree from Tsinghua University, Beijing, China, in 2012. His current research interests include image and video editing, and computer graphics.



Kun Xu is an associate professor in the Department of Computer Science and Technology, Tsinghua University. He received his Ph. D. degree from Tsinghua University in 2009. His research interests include realistic rendering, and image and video editing.

Ralph R. Martin obtained a Ph.D. degree from Cambridge University, U.K. in 1983. He has been at Cardiff University, U.K., since 1982, where he has been a Professor since 2000 and currently leads the Visual Computing Research Group. He is a Guest Professor at Tsinghua University and other universities in China. His current research interests include modeling, reverse engineering, intelligent sketch input, mesh processing, video processing, vision-based geometric inspection, and geometric reasoning. He is a Fellow of: the Learned Society of Wales, the Institute of Mathematics and its Applications, and the British Computer Society. He is is Associate Editor-in-Chief of Computational Visual Media, on the editorial boards of several other journals, including Computer Aided Design, Computer Aided Geometric Design, Computers & Graphics, and Geometric Models.

Fei-Yue Huang received his Ph. D. degree from Tsinghua University, China, in 2008. He is currently Director of the Social Network Platform Department of Tencent. His research interests include face recognition, pattern recognition and machine learning.

Shi-Min Hu is currently a professor in the Department of Computer Science and Technology, Tsinghua University, China. He received the PhD degree from Zhejiang University in 1996. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He has published more than 100 papers in journals and refereed conference. He is Editor-in-Chief of Computational Visual media, and on editorial board of several journals, including IEEE Transactions on Visualization and Computer Graphics, Computer Aided Design and Computer & Graphics.