Colour quantisation using the adaptive distributing units algorithm

M. E. Celebi*[1], S. Hwang[2] and Q. Wen[3]

Colour quantisation (CQ) is an important operation with many applications in graphics and image processing. Most CQ methods are essentially based on data clustering algorithms one of which is the popular k-means algorithm. Unfortunately, like many batch clustering algorithms, k-means is highly sensitive to the selection of the initial cluster centres. In this paper, we adapt Uchiyama and Arbib's competitive learning algorithm to the CQ problem. In contrast to the batch k-means algorithm, this online clustering algorithm does not require cluster centre initialisation. Experiments on a diverse set of publicly available images demonstrate that the presented method outperforms some of the most popular quantisers in the literature.

Keywords: Colour quantization, Clustering, K-means, Competitive learning

Introduction

True-colour images typically contain thousands of colours, which makes their display, storage, transmission, and processing problematic. For this reason, CQ is commonly used as a pre-processing step for various graphics and image processing tasks. In the past, CQ was a necessity due to the limitations of the display hardware, which could not handle over 16 million possible colours in 24-bit images. Although 24-bit display hardware has become more common, CQ still maintains its practical value.[1] Modern applications of CQ in graphics and image processing include: compression,[2] segmentation,[3] text localisation/detection,[4] colour–texture analysis,[5] watermarking,[6] non-photorealistic rendering,[7] and content-based retrieval.[8]

The process of CQ is mainly comprised of two phases: palette design (the selection of a small set of colours that represents the original image colours) and pixel mapping (the assignment of each input pixel to one of the palette colours). The primary objective is to reduce the number of unique colours, N', in an image to K ($K \ll N'$) with minimal distortion. In most applications, 24-bit pixels in the original image are reduced to 8 bits or fewer. Since natural images often contain a large number of colours, faithful representation of these images with a limited size palette is a difficult problem.

CQ methods can be broadly classified into two categories:[9] image-independent methods that determine a universal (fixed) palette without regard to any specific image,[10,11] and image-dependent methods that determine a custom (adaptive) palette based on the colour distribution of the images. Despite being very fast, image-independent methods usually give poor results since they do not take into account the image contents. Therefore, most of the studies in the literature consider only image-dependent methods, which strive to achieve a better balance between computational efficiency and visual quality of the quantisation output.

Numerous image-dependent CQ methods have been developed over the past three decades. These can be categorised into two families: pre-clustering methods and post-clustering methods.[1] Pre-clustering methods are mostly based on the statistical analysis of the colour distribution of the images. Divisive pre-clustering methods start with a single cluster that contains all N' image colours. This initial cluster is recursively subdivided until K clusters are obtained. Well-known divisive methods include median-cut,[12] octree,[13] variance-based method,[14] binary splitting,[15] greedy orthogonal bipartitioning,[16] centre-cut,[17] and rwm-cut.[18] More recent methods can be found elsewhere.[19–23] On the other hand, agglomerative pre-clustering methods[24–28] start with N' singleton clusters each of which contains one image colour. These clusters are repeatedly merged until K clusters remain. In contrast to pre-clustering methods that compute the palette only once, post-clustering methods first determine an initial palette and then improve it iteratively. Essentially, any data clustering method can be used for this purpose. Since these methods involve iterative or stochastic optimisation, they can obtain higher quality results when compared to pre-clustering methods at the expense of increased computational time. Clustering algorithms adapted to CQ include maxmin,[29,30] k-means,[31–35] k-harmonic means,[36] competitive learning,[37,38] fuzzy c-means,[39–45] rough c-means,[46] BIRCH,[47] and self-organising maps.[48–53]

In this paper, we adapt Uchiyama and Arbib's adaptive distributing units (ADU) algorithm[54] to the CQ problem. The rest of the paper is organised as

[1]Department of Computer Science, Louisiana State University, Shreveport, LA, USA
[2]Department of Computer Science, University of Illinois, Springfield, IL, USA
[3]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

*Corresponding author: M. E. Celebi, Department of Computer Science, Louisiana State University, Shreveport, LA, USA; email: ecelebi@lsus.edu

Received 16 February 2012; accepted 30 July 2013
DOI 10.1179/1743131X13Y.0000000059

follows. The section on 'BATCH VS ONLINE CLUSTERING ALGORITHMS' describes the conventional batch k-means and ADU algorithms. The section on 'EXPERIMENTAL RESULTS AND DISCUSSION' presents the experimental setup and compares the proposed method to other CQ methods. Finally, the section on 'CONCLUSION' gives the conclusions.

Batch vs online clustering algorithms

Batch k-means algorithm

Given a data set $\mathcal{X}=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N\}\in\mathbb{R}^D$, hard partitional algorithms divide $\mathcal{X}$ into K exhaustive and mutually exclusive clusters $\mathcal{S}=\{S_1,S_2,\ldots,S_k\}$, $\bigcup_{k=1}^{K} S_k=\mathcal{X}$, $S_i\cap S_j=\varnothing$ for $1\leq i\neq j\leq K$. These algorithms usually generate clusters by optimising a criterion function. The most intuitive and frequently used criterion function is the sum of squared error (SSE) given by

$$SSE=\sum_{k=1}^{K}\sum_{\mathbf{x_i}\in S_k}\|\mathbf{x_i}-\mathbf{c_k}\|_2^2 \tag{1}$$

where $\|\cdot\|_2$ denotes the Euclidean ($\mathcal{L}_2$) norm and $\mathbf{c}_i=1/|S_i|\sum_{\mathbf{x}_j\in S_i}\mathbf{x_j}$ is the centroid of cluster S_i whose cardinality is $|S_i|$.

The number of ways in which a set of N objects can be partitioned into K non-empty groups is given by Stirling numbers of the second kind

$$\left\{ {N \atop K} \right\}=\frac{1}{K!}\sum_{i=0}^{K}(-1)^{K-i}\binom{K}{i}i^N \tag{2}$$

which can be approximated by $K^N/K!$. It can be seen that a complete enumeration of all possible clusterings to determine the global minimum of (1) is clearly computationally prohibitive except for very small data sets. In fact, this non-convex optimisation problem is proven to be NP-hard even for $K=2$ (Ref. 55) or $D=2$ (Ref. 56). Consequently, various heuristics have been developed to provide approximate solutions to this problem.[57] Among these heuristics, Lloyd's algorithm,[58] often referred to as the *batch k-means algorithm*, is the simplest and most commonly used one. This algorithm starts with K arbitrary centres, typically chosen uniformly at random from the data points. Each point is assigned to the nearest centre and then each centre is recalculated as the mean of all points assigned to it. These two steps are repeated until a predefined termination criterion is met. The pseudocode for this procedure is given in Algorithm (1) (**bold** symbols denote vectors). Here $m[i]$ denotes the membership of point $\mathbf{x_i}$, i.e. index of the cluster centre that is nearest to $\mathbf{x_i}$

Algorithm 1: batch k-means algorithm

input $\mathcal{X}=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x_N}\}\in\mathbb{R}^D$
($N\times D$ input data set)
output $\mathcal{C}=\{\mathbf{c}_1,\mathbf{c}_2,\ldots,\mathbf{c_K}\}\in\mathbb{R}^D$ (K cluster centres)
Select a random subset $\mathcal{C}$ of $\mathcal{X}$ as the initial set of cluster centres
while *termination criterion is not met* **do**
  for $(i=1;i\leq N;i=i+1)$ do
    Assign $\mathbf{x_i}$ to the nearest cluster;
    $m[i]=\underset{k\in\{1,2,\ldots,K\}}{argmin}\ \|\mathbf{x_i}-\mathbf{c_k}\|^2$
  end
  Recalculate the cluster centres;
  for $(k=1;k\leq K;k=k+1)$ do
    Cluster S_k contains the set of points $\mathbf{x_i}$ that are nearest to the centre $\mathbf{c_k}$
    $S_k=\{\mathbf{x_i}|m[i]=k\}$
    Calculate the new centre $\mathbf{c_k}$ as the mean of points that belong to S_k
    $\mathbf{c_k}=\frac{1}{|S_k|}\sum_{\mathbf{x_i}\in S_k}\mathbf{x_i}$
  end
end

From a CQ perspective the batch k-means algorithm has two main drawbacks. First, due to its batch nature, i.e. cluster centres are updated after the presentation of all input vectors, the algorithm might get stuck in a local minimum. Second, the algorithm is highly sensitive to the selection of the initial cluster centres.[59] Adverse effects of improper initialisation include empty clusters (dead units), slower convergence, and a higher chance of getting stuck in bad local minima.[35]

Adaptive distributing units (ADU) algorithm

The ADU algorithm is an online clustering algorithm based on the competitive learning paradigm, which is closely related to neural networks.[60] According to Rumelhart and Zipser,[61] a competitive learning scheme consists of the following three basic components:

1. Start with a set of units that are all the same except for some randomly distributed parameter, which makes each of them respond slightly differently to a set of input patterns.
2. Limit the 'strength' of each unit.
3. Allow the units to compete in some way for the right to respond to a given subset of inputs.

The pseudocode for the ADU algorithm is given in Algorithm (2). Here, at any given time, n and $wc[i]$ denote the number of units (clusters) determined so far and the number of times that the i-th unit, represented by $\mathbf{c_i}$, won the competition in the past, respectively. The algorithm parameters θ, t_{max}, and γ denote the maximum number of times a particular unit can win, maximum number of iterations, and the learning rate, respectively. The procedure starts with a single unit whose centre, $\mathbf{c}_1$, is given by the centroid of the data set $\mathcal{X}$. In each iteration, an input vector $\mathbf{x}$ is randomly selected from $\mathcal{X}$ and the unit that is nearest to $\mathbf{x}$ with respect to Euclidean distance is declared as the winner. This unit is then updated by moving its centre closer to $\mathbf{x}$ and incrementing its win count. New units are added by splitting existing units that reach the threshold number of wins, θ, until the number of units reaches K. This splitting rule prevents certain units from monopolising the input vectors and thus effectively avoids the dead unit problem. When a new unit is generated by splitting an existing unit, the two units are temporarily colocated. Whenever an input vector that is nearest to these two units is presented, one of them becomes the winner and gets updated, while the other one remains unchanged. Hence, the problem of two units moving together cannot occur. Following,[54] the algorithm parameters were set to $\theta=400K^{1/2}$, $t_{max}=(2K-3)\theta$, and $\gamma=0{\cdot}015$.

Algorithm 2: adaptive distributing units algorithm

input $\mathcal{X}=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x_N}\}\in\mathbb{R}^D$ ($N\times D$ input data set)
output $\mathcal{C}=\{\mathbf{c}_1,\mathbf{c}_2,\ldots,\mathbf{c_K}\}\in\mathbb{R}^D$ (K cluster centres)

The first unit is given by the centroid of the input vectors
$n=1$

$\mathbf{c_n}=1/N\sum_{i=1}^{N}\mathbf{x_i}$

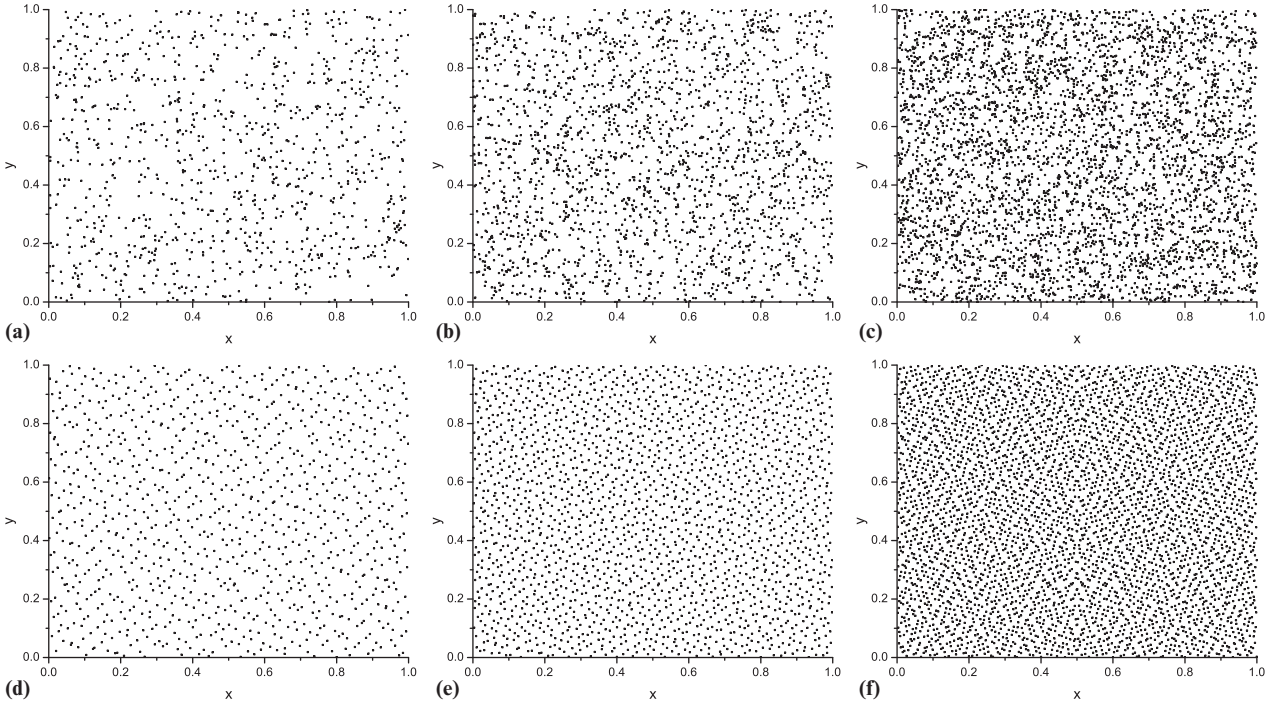
Initialise the win counts to 0;
for $(i=1; i\leq K; i=i+1)$ do
  $wc[i]=0$;
end
for $(t=1; t\leq t_{max}; t=t+1)$ do
  Select an input vector $\mathbf{x}$ randomly from $\mathcal{X}$
  Determine the winner (nearest) unit;
  $winner=\underset{k\in\{1,2,\ldots,n\}}{argmin}\ \|\mathbf{x}-\mathbf{c_k}\|^2$
  Update the winner and its win count;
  $\mathbf{c_{winner}}=\mathbf{c_{winner}}+\gamma(\mathbf{x}-\mathbf{c_{winner}})$;
  $wc[winner]=wc[winner]+1$;
  Split the unit if its win count exceeds the threshold;
  if $wc[winner]=\theta$ and $n<K$ **then**
    $n=n+1$
    $\mathbf{c_n}=\mathbf{c_{winner}}$;
    $wc[n]=wc[winner]=0$;
  end
end

Due to their online formulation, i.e. cluster centres are updated after the presentation of each input vector, clustering algorithms based on the competitive learning paradigm are generally more adaptive and thus more likely to escape local minima when compared to batch algorithms. The online nature of such clustering algorithms, however, presents two drawbacks. First, these algorithms are order-dependent, that is, different presentation orders of the data vectors induce different partitions. Second, random selection of the input vectors renders these algorithms nondeterministic, i.e. each run could potentially generate different clustering results. We avoid these problems by substituting the *pseudo-random* sampling scheme used in the ADU algorithm with *quasi-random* sampling. More specifically, we sample the image data by means of a quasi-random Sobol' sequence.[62] A quasi-random sequence is a sequence of D-tuples that fill $\mathbb{R}^D$ more uniformly than uncorrelated pseudo-random points.[63] This is illustrated in Fig. 1. Here the top row shows three pseudo-random sequences with increasing length from left to right generated by the MT19937 variant of the celebrated Mersenne Twister algorithm,[64] whereas the bottom row shows the corresponding quasi-random sequences generated by a Sobol' sequence. It can be seen that the pseudo-random sequences exhibit clumping, leading to rather uneven coverage of the sampled area,[65] whereas the quasi-random sequences result in a significantly more uniform point distribution. It should be noted that, despite their name, quasi-random sequences are in fact completely deterministic. This means that the proposed sampling scheme selects exactly the same set of pixels in each run. Since the theory and implementation of Sobol' sequences is mathematically involved, the interested reader is referred to the relevant literature[62,63,65] for further information.

Experimental results and discussion

Image set and performance criteria

The proposed method was tested on a set of eight true-colour (24-bit) test images commonly used in the quantisation literature: Baboon (USC-SIPI Image Database, 512×512, 230427 colours), Flowers & Sill (Kodak Lossless True Colour Image Suite, 768×512, 37552 colours), Hats (Kodak Lossless True Colour Image Suite, 768×512, 34871 colours), Lenna (USC-SIPI Image Database, 512×512, 148279 colours),



1 Comparison of pseudo-random *a, b, c* and quasi-random sampling *d, e, f*: *a* random sequence (1024 pts); *b* random sequence (2048 pts); *c* random sequence (4096 pts); *d* Sobol' sequence (1024 pts); *e* Sobol' sequence (2048 pts); *f* Sobol' sequence (4096 pts)

Motocross (Kodak Lossless True Colour Image Suite, 768×512, 63558 colours), Parrots (Kodak Lossless True Colour Image Suite, 768×512, 72079 colours), Pills (Karel de Gendre, 800×519, 206609 colours), and Sailboats (Kodak Lossless True Colour Image Suite, 512×768, 24106 colours). These images are shown in Fig. 2.

The effectiveness of a quantisation method was quantified by the mean squared error (MSE) measure

$$MSE\left(\mathbf{X},\hat{\mathbf{X}}\right) = \frac{1}{HW}\sum_{\mathrm{h}=1}^{\mathrm{H}}\sum_{\mathrm{w}=1}^{\mathrm{W}}\left\|\mathbf{X}(h,w)-\hat{\mathbf{X}}(h,w)\right\|_2^2 \qquad (3)$$

where $\mathbf{X}$ and $\hat{\mathbf{X}}$ denote respectively the $H \times W$ original and quantised images in the RGB colour space. MSE represents the average colour distortion with respect to the $\mathcal{L}_2^2$ (squared Euclidean) norm and is the most commonly used evaluation measure in the quantisation literature.[1]

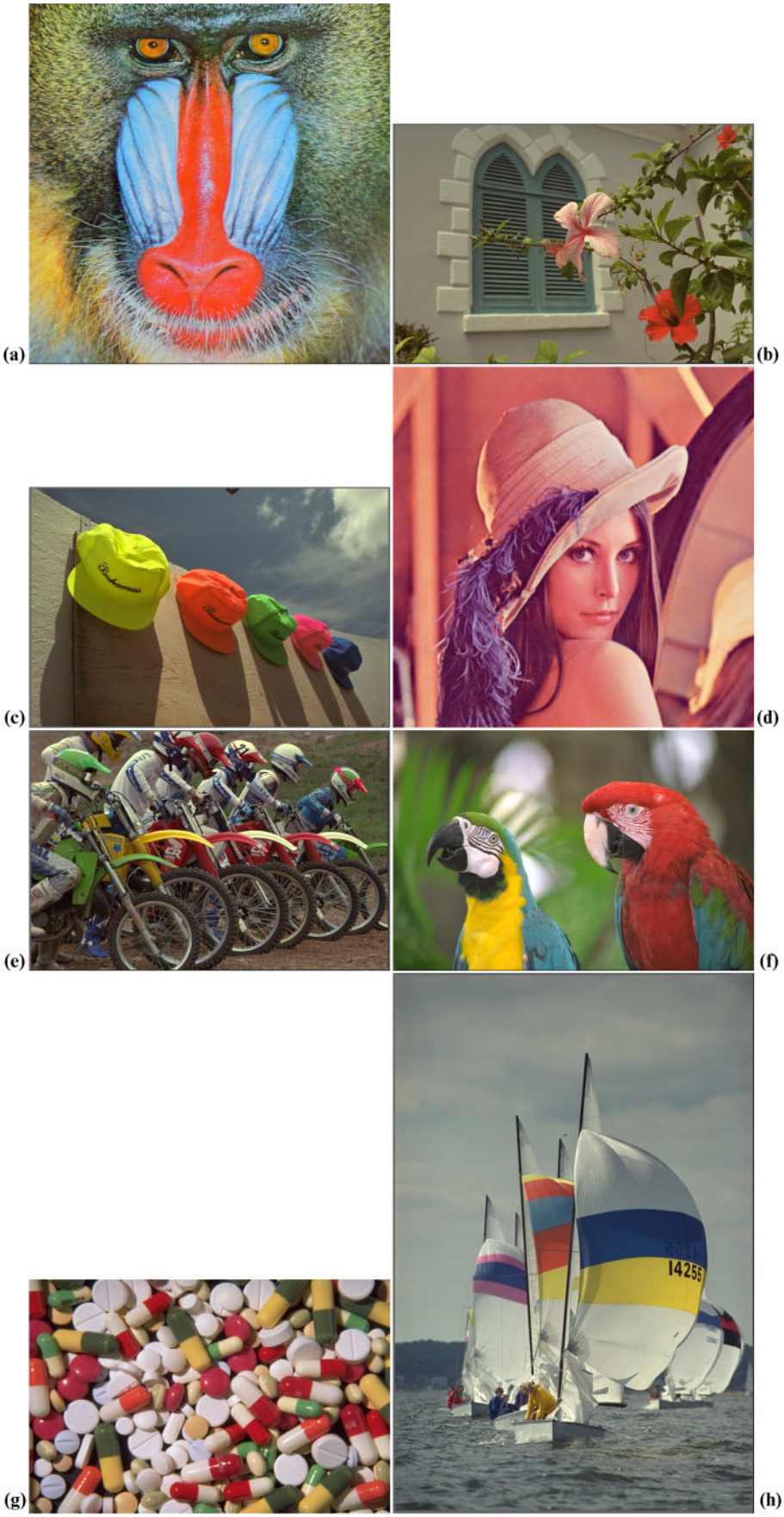
Comparison of ADU against other CQ methods

The proposed method was compared to 13 well-known CQ methods:

- **Popularity (POP)**:[12] This method builds a $16 \times 16 \times 16$ colour histogram using 4 bits/channel uniform quantisation and then takes the K most frequent colours in the histogram as the colour palette.
- **Median-cut (MC)**:[12] This method starts by building a $32 \times 32 \times 32$ colour histogram using uniform quantisation. This histogram volume is then recursively split into smaller boxes until K boxes are obtained. At each step, the box that contains the greatest number of colours is split along the longest axis at the median point, so that the resulting subboxes each contain approximately the same number of colours. The centroids of the final K boxes are taken as the colour palette.
- **Modified Popularity (MPOP)**:[66] This method starts by building a $2^{\mathrm{R}} \times 2^{\mathrm{R}} \times 2^{\mathrm{R}}$ colour histogram using R bits/channel uniform quantisation. It chooses the most frequent colour as the first palette colour $\mathbf{c_1}$ and then reduces the frequency of each colour $\mathbf{c}$ by a factor of $\left(1-e^{\alpha\|\mathbf{c}-\mathbf{c}_1\|_2^2}\right)$, where α is a user-defined parameter. The remaining palette colours are chosen similarly. In the experiments, best results were obtained with the following parameter configuration: $\alpha=0{\cdot}25$ and $R=4$ for $K \leq 64$ and $R=5$ otherwise.
- **Octree (OCT)**:[13] This two-phase method first builds an octree (a tree data structure in which each internal node has up to eight children) that represents the colour distribution of the input image and then, starting from the bottom of the tree, prunes the tree by merging its nodes until K colours are obtained. In the experiments, the tree depth was limited to 6 to obtain the best results.
- **Variance-based method (WAN)**:[14] This method is similar to MC with the exception that at each step the box with the greatest SSE is split along the axis with the least weighted sum of projected variances at the point that minimises the marginal squared error.
- **Greedy orthogonal bipartitioning (WU)**:[16] This method is similar to WAN with the exception that at each step the box is split along the axis that minimises the sum of variances on both sides.
- **Centre-cut (CC)**:[17] This method is similar to MC with the exception that at each step the box with the greatest range on any coordinate axis is split along its longest axis at the mean point.
- **Self-organising map (SOM)**:[48] This method utilises a one-dimensional self-organising map with K neurons. A random subset of N/f pixels is used in the training phase and the final weights of the neurons are taken as the colour palette. In the experiments, the sampling factor was set to $f=1$ to obtain the best results.
- **Radius-weighted mean-cut (RWM)**:[18] This method is similar to WAN with the exception that the box is split along the vector from the origin to the radius-weighted mean (rwm) at the rwm point.
- **Modified maxmin (MMM)**:[30] This method chooses the first palette colour $\mathbf{c_1}$ arbitrarily from the input image colours and the i-th colour $\mathbf{c_i}$ ($i=2,3,\ldots,K$) is chosen to be the colour that has the greatest minimum weighted $\mathcal{L}_2^2$ distance (the weights for the red, green, and blue channels are taken as 0·5, 1·0, and 0·25, respectively) to the previously selected colours, i.e. $\mathbf{c}_1,\mathbf{c}_2,\ldots,\mathbf{c}_{\mathbf{i}-1}$. Each of these initial palette colours is then recalculated as the mean of the colours assigned to it. In the experiments, the first colour was chosen as the centroid of the input image colours.
- **Pairwise clustering (PWC)**:[27] This method is an adaptation of Ward's agglomerative hierarchical clustering method[67] to CQ. It builds a $2^{\mathrm{R}} \times 2^{\mathbf{R}} \times 2^{\mathbf{R}}$ colour histogram and constructs a $Q \times Q$ joint quantisation error matrix, where Q is the number of colours in the reduced colour histogram. The clustering procedure starts with Q singleton clusters each of which contains one image colour. In each iteration, the pair of clusters with the least joint quantisation error are merged. This merging process is repeated until K clusters remain.
- **Split and Merge (SAM)**:[28] This two-phase method first partitions the colour space uniformly into B partitions. This initial set of B clusters is represented as an adjacency graph. In the second phase, $(B-K)$ merge operations are performed to obtain the final K clusters. At each step of the second phase, the pair of clusters with the least joint quantisation error are merged. In the experiments, the initial number of clusters was set to $B=20K$ to obtain the best results.
- **Cheng and Yang (CY)**:[20] This method is similar to WAN with the exception that at each step the box is split along a specially chosen line defined by the mean colour and the colour that is farthest away from it at the mean point.

Two variants of ADU were implemented: one with pseudo-random sampling (ADU-PRS) and the other with quasi-random sampling (ADU-QRS). We also implemented the batch k-means (KM) algorithm with randomly chosen initial centres. Convergence of KM was determined by the following commonly used criterion: $(SSE_{\mathrm{i}-1}-SSE_{\mathrm{i}})/SSE_{\mathrm{i}} \leq \varepsilon$, where SSE_{i} denotes the SSE (1) value at the end of the i-th iteration. The convergence threshold was set to $\varepsilon=0{\cdot}001$.

Table 1 compares the effectiveness of the CQ methods on the test images. The best (lowest) error values are shown in **bold**. Note that since ADU-PRS and KM involve randomness, the quantisation errors for these methods are specified in the form of mean $\pm$ standard deviation over 100 runs. The following observations are in order:

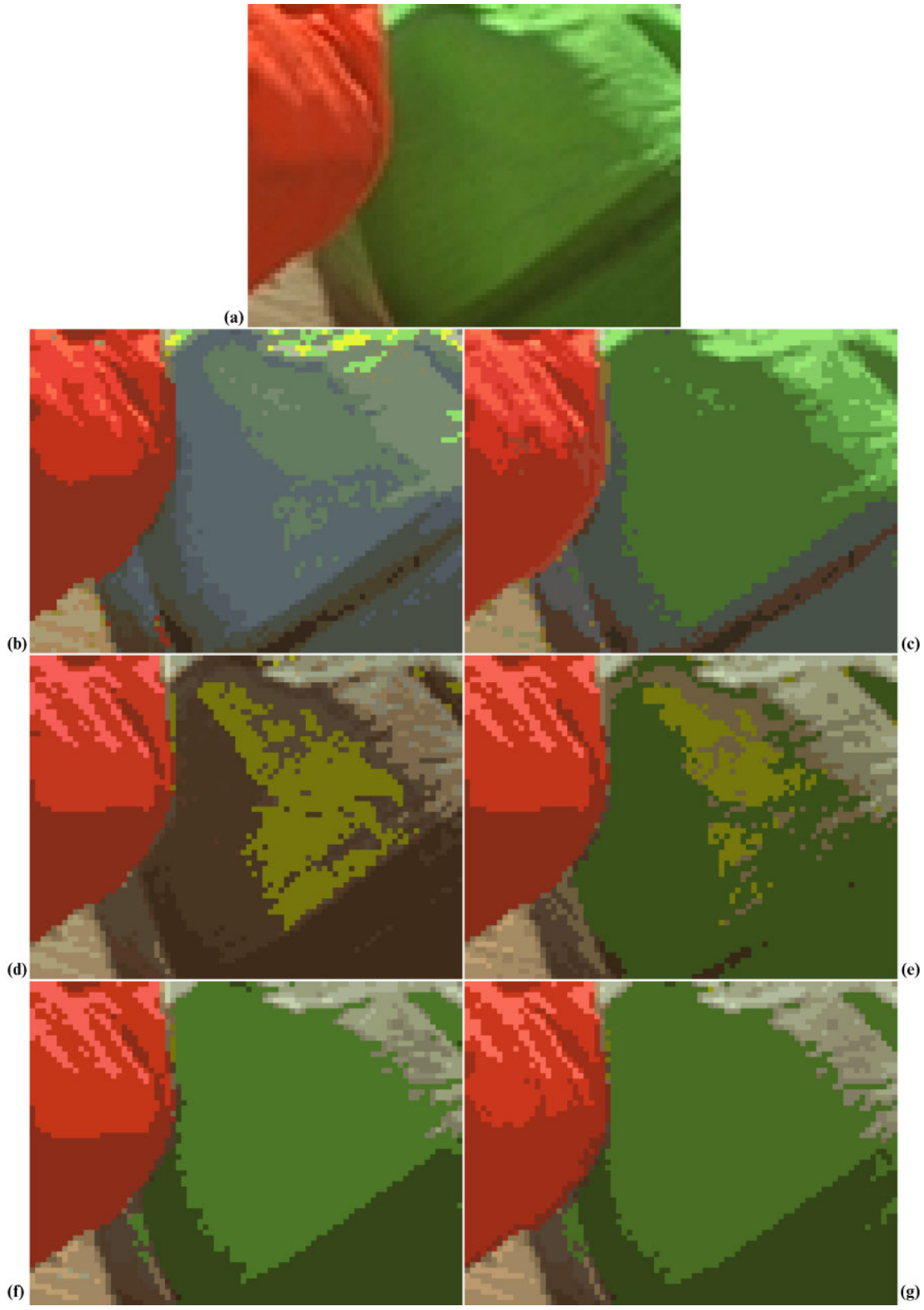


2 Test images: *a* Baboon; *b* Flowers & Sill; *c* Hats; *d* Lenna; *e* Motocross; *f* Parrots; *g* Pills; *h* Sailboats

Table 1 MSE comparison of the quantisation methods

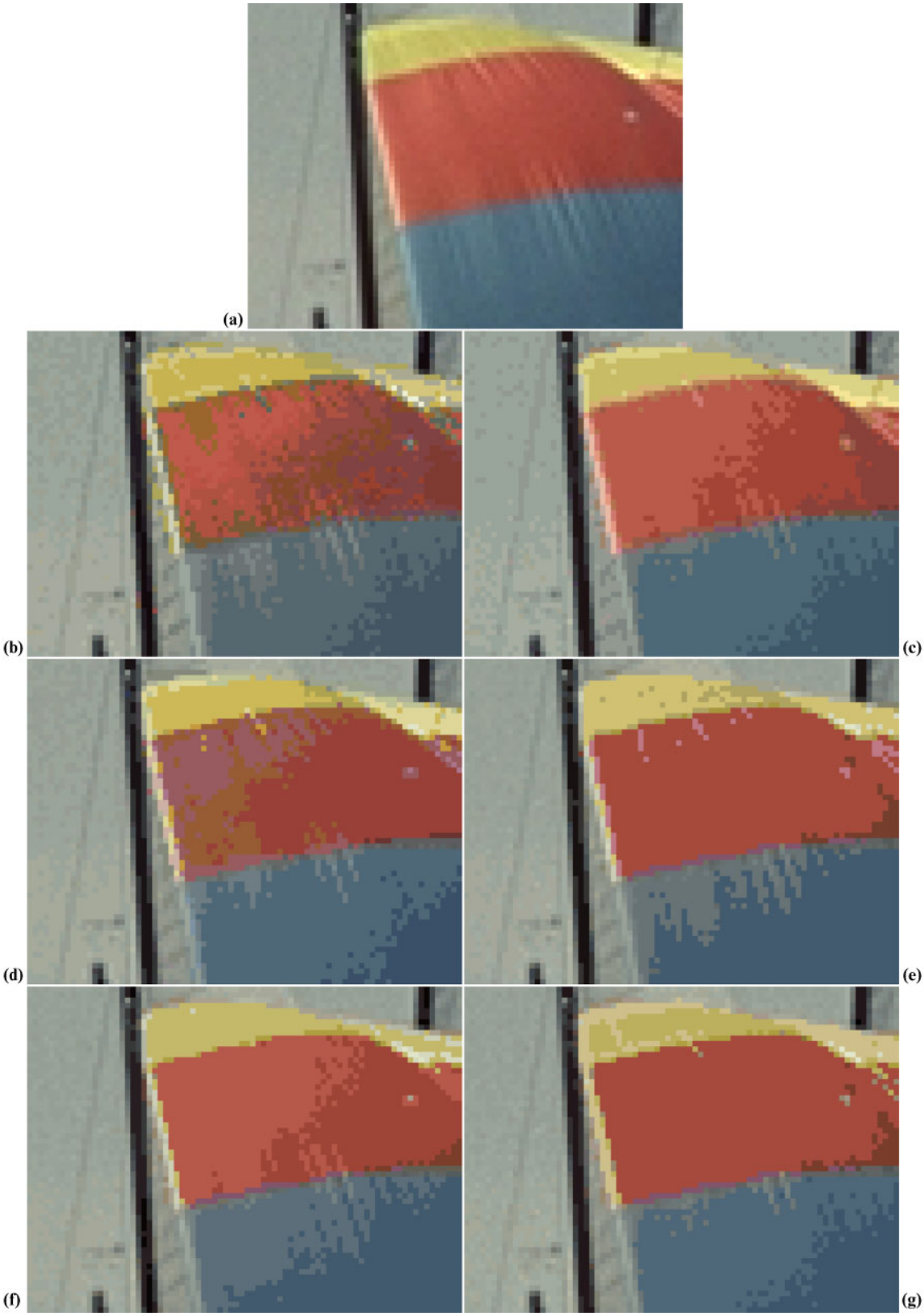
	K				K				K				K			
	32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
Method	Baboon				Flowers & Sill				Hats				Lenna			
POP	1680	849	331	170	523	364	84	65	1240	463	129	76	347	199	84	65
MC	643	446	307	213	271	143	93	56	588	207	123	77	214	146	112	80
MPOP	453	290	195	109	161	112	43	32	244	137	59	37	195	139	60	48
OCT	530	307	204	125	155	88	47	26	293	132	65	34	187	110	66	41
WAN	528	386	266	178	198	98	57	37	624	311	106	53	217	141	88	57
WU	468	288	187	119	134	71	40	26	213	103	52	30	158	99	62	39
CC	473	300	203	145	206	113	70	40	284	153	89	52	189	126	81	52
SOM	434	269	164	108	189	80	35	22	214	96	45	28	140	87	50	34
RWM	459	302	188	120	149	75	41	28	210	105	58	37	161	95	60	39
MMM	510	368	230	147	175	119	58	35	282	142	93	51	183	114	73	48
PWC	461	308	207	129	159	100	73	64	239	135	93	72	187	108	79	65
SAM	465	294	189	120	141	85	48	31	180	95	54	33	158	102	65	45
CY	466	281	187	118	141	78	43	29	247	124	67	39	166	98	63	42
KM	381±4	239±2	153±1	98±1	109±7	61±3	36±1	22±1	190±17	97±12	50±3	28±2	121±2	74±1	47±1	31±0
ADU-PRS	383±2	238±1	152±0	97±0	106±2	59±1	33±0	20±0	170±6	88±3	46±1	25±0	121±1	73±0	46±0	30±0
ADU-QRS	380	237	152	97	103	59	33	19	169	86	47	25	120	73	46	30

	K				K				K				K			
	32	64	128	256	32	64	128	256	32	64	128	256	32	64	128	256
Method	Motocross				Parrots				Pills				Sailboats			
POP	1289	474	202	93	4087	372	181	104	788	223	124	85	397	107	75	68
MC	438	254	169	114	441	265	154	112	324	234	159	100	127	92	66	52
MPOP	288	178	84	53	380	212	105	59	277	175	88	55	126	89	33	26
OCT	301	159	96	54	342	191	111	64	282	160	99	57	127	72	36	19
WAN	446	292	169	92	376	233	153	92	295	198	133	88	148	83	49	32
WU	268	147	87	51	299	167	95	58	261	150	89	55	86	50	29	19
CC	335	202	123	75	399	247	149	79	286	172	112	77	107	69	48	32
SOM	302	135	70	44	279	152	82	48	226	138	72	46	126	58	26	18
RWM	251	150	84	51	297	171	100	61	260	150	89	56	92	51	33	23
MMM	408	277	138	86	352	195	129	68	276	175	117	76	171	78	51	30
PWC	243	161	102	78	349	205	126	86	270	171	112	76	130	90	71	66
SAM	238	138	82	53	282	158	92	59	246	141	85	54	107	63	38	28
CY	248	147	89	53	313	179	107	65	238	158	96	59	110	56	36	25
KM	200±5	115±2	70±2	43±1	245±6	135±3	78±2	47±1	206±7	115±2	68±1	43±1	85±12	47±4	27±2	16±1
ADU-PRS	200±3	113±3	67±1	40±0	241±4	132±1	75±1	44±0	203±1	114±1	67±0	41±0	87±2	46±2	26±1	15±0
ADU-QRS	205	110	67	40	237	130	74	44	201	113	67	41	82	44	26	15



3 Hats output images (K=32): *a* Original; *b* WAN output; *c* MC output; *d* SOM output; *e* KM output; *f* ADU-PRS output; *g* ADU-QRS output

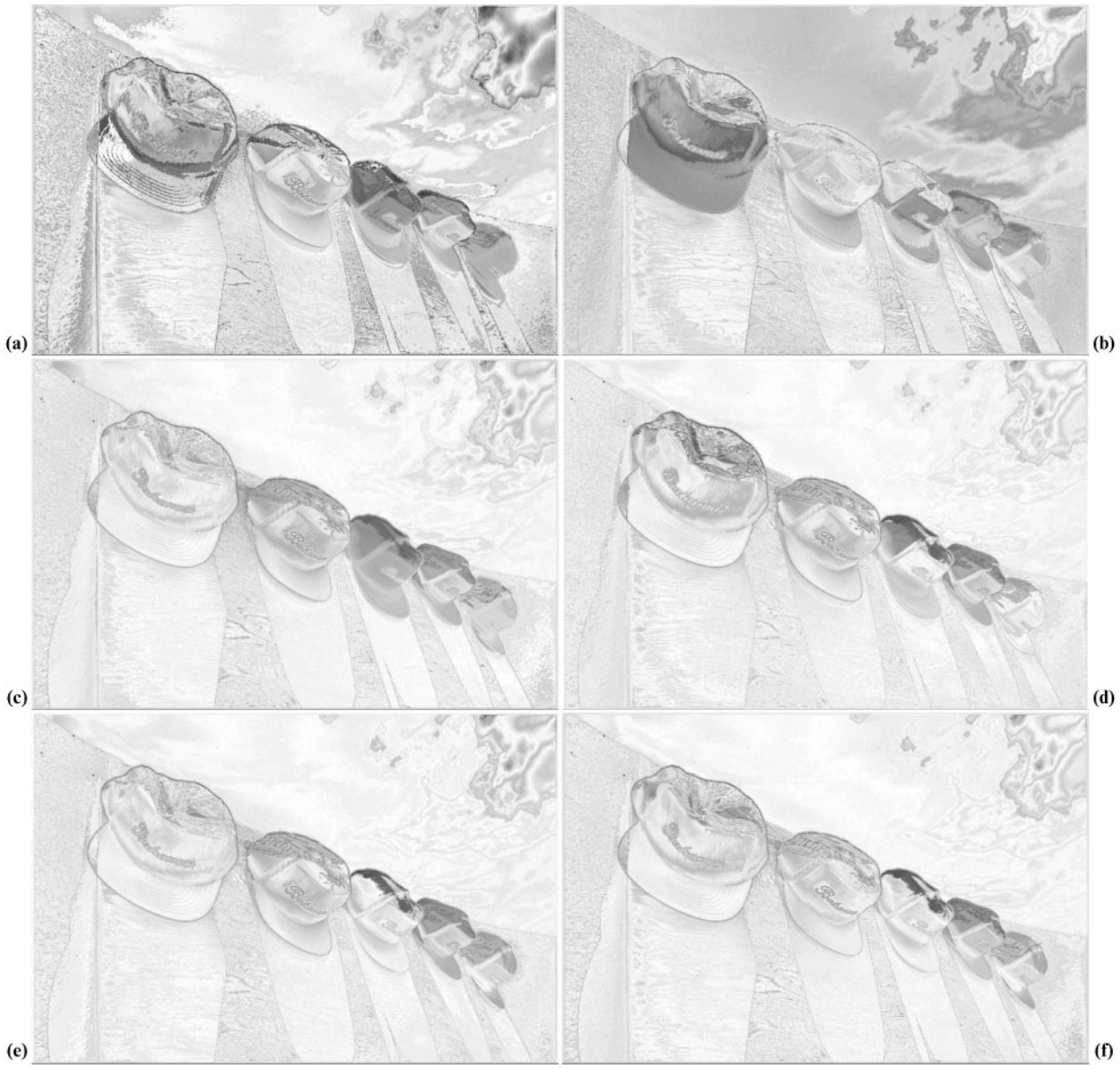
- In general, post-clustering methods, i.e. SOM, MMM, KM, ADU-PRS, and ADU-QRS, are significantly more effective than the pre-clustering methods.
- KM, ADU-PRS, and ADU-QRS are the most effective post-clustering methods.
- Except in few cases, ADU-PRS is slightly more effective than KM. Furthermore, as expected, ADU-PRS is significantly less sensitive to randomness, which is evidenced by its low standard deviation.
- In about half of the cases, ADU-QRS is more effective than ADU-PRS, whereas in the remaining



4 Sailboats output images (K=64): *a* Original; *b* WAN output; *c* MMM output; *d* OCT output; *e* KM output;*f* ADU-PRS output; *g* ADU-QRS output

cases the two methods have similar effectiveness. Note that the primary advantage of ADU-QRS over KM and ADU-PRS is its deterministic formulation, i.e. it gives exactly the same results in every run.

- POP is the least effective pre-clustering method. This is not surprising as this method disregards colours in sparse regions of the colour space. Interestingly, despite being a simple modification of POP, MPOP performs surprisingly well, surpassing some of the better known methods such as MC, OCT, WAN, and CC.
- Despite its iterative nature, MMM performs poorly even when compared to pre-clustering methods. This



5 Hats error images (K=32): *a* WAN error; *b* MC error; *c* SOM error; *d* KM error; *e* ADU-PRS error; *f* ADU-QRS error

is because MMM tries to distribute the quantisation distortion more or less evenly throughout the image at the expense of increased mean distortion.

Figures 3 and 4 show sample quantisation results for close-up parts of the Hats and Sailboats images, respectively. Figures 5 and 6 show the full-scale error images for the respective images. The error image for a particular quantisation method was obtained by taking the pixelwise absolute difference between the original and quantised images. In order to obtain a better visualisation, pixel values of the error images were multiplied by 4 and then negated. It can be seen that the proposed method, ADU-QRS, performs exceptionally well in allocating representative colours to various image regions, resulting in cleaner error images.

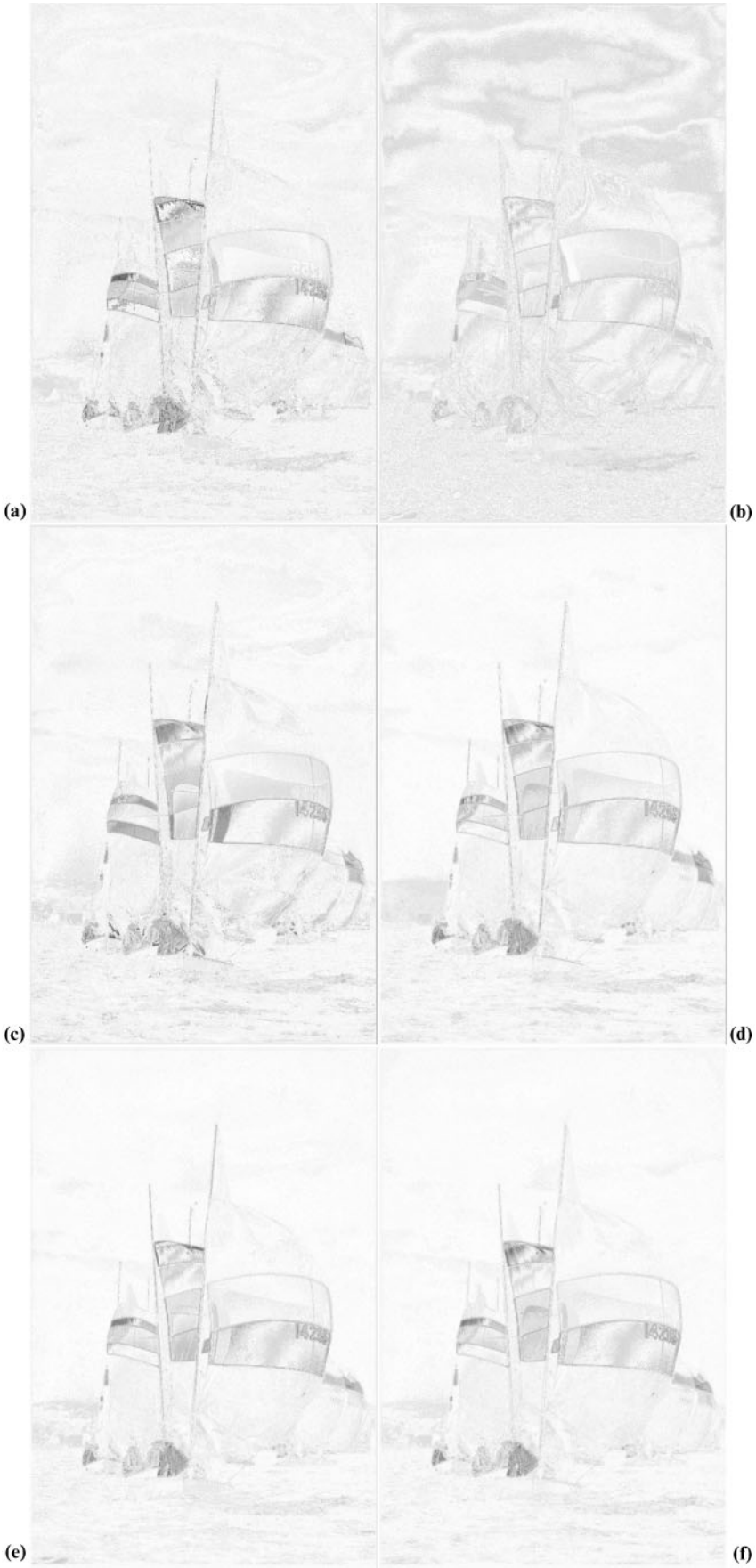
Conclusions

In this paper, an effective CQ method based on the ADU algorithm was introduced. This online clustering algorithm is advantageous over the conventional batch k-means clustering algorithm in that it does not require cluster centre initialisation and that it avoids the dead unit problem. Due to its online formulation, however, the ADU algorithm is both order-dependent and nondeterministic. These drawbacks were overcome using quasi-random sampling. The proposed ADU variant with quasi-random sampling, ADU-QRS, was shown to give comparable or better results than the original one based on pseudo-random sampling, ADU-PRS.

Experiments on a diverse set of classic test images demonstrated that the proposed method outperforms well-known CQ methods with respect to distortion minimisation. In addition, our method is significantly easier to implement when compared to dedicated CQ methods.

Acknowledgements

This publication was made possible by grants from the Louisiana Board of Regents (LEQSF2008-11-RD-A-12), US National Science Foundation (0959583, 1117457), and National Natural Science Foundation of China (61050110449, 61073120).



6 Sailboats error images (K=64): *a* WAN error; *b* MMM error; *c* OCT error; *d* KM error; *e* ADU-PRS error; *f* ADU-QRS error

References

1. L. Brun, A. Trémeau, Digital Color Imaging Handbook, 2002, Ch. Color Quantization, pp. 589–638 (CRC Press, Boca Raton, FL).
2. Yang, C.-K. and Tsai, W.-H. Color image compression using quantization, thresholding, and edge detection techniques all based on the moment-preserving principle. *Pattern Recognit. Lett*., 1998, **19**, (2), 205–215.
3. Deng, Y. and Manjunath, B. Unsupervised segmentation of color-texture regions in images and video. *IEEE Trans. Pattern Anal. Machine Intell*., 2001, **23**, (8), 800–810.
4. Sherkat, N., Allen, T. and Wong, S. Use of colour for hand-filled form analysis and recognition. *Pattern Anal. Appl*., 2005, **8**, (1), 163–180.
5. Sertel, O., Kong, J., Catalyurek, U. V., Lozanski, G., Saltz, J. H. and Gurcan, M. N. Histopathological image analysis using model-based intermediate representations and color texture: follicular lymphoma grading. *J. Signal Process. Syst*., 2009, **55**, (1–3), 169–183.
6. Kuo, C.-T. and Cheng, S.-C. Fusion of color edge detection and color quantization for color image watermarking using principal axes analysis. *Pattern Recognit*., 2007, **40**, (12), 3691–3704.
7. Wang, S., Cai, K., Lu, J., Liu, X. and Wu, E. Real-time coherent stylization for augmented reality. *Visual Comput*., 2010, **26**, (6–8), 445–455.
8. Deng, Y., Manjunath, B., Kenney, C., Moore, M. and Shin, H. An efficient color representation for image retrieval. *IEEE Trans. Image Process*., 2001, **10**, (1), 140–147.
9. Xiang, Z. Handbook of Approximation Algorithms and Metaheuristics, 2007, Ch. Color Quantization, pp. 86-1–86-17 (Chapman & Hall/CRC, London).
10. Gentile, R. S., Allebach, J. P. and Walowit, E. Quantization of color images based on uniform color spaces. *J. Imag. Technol*., 1990, **16**, (1), 11–21.
11. Mojsilovic, A. and Soljanin, E. Color quantization and processing by Fibonacci lattices. *IEEE Trans. Image Process*., 2001, **10**, (11), 1712–1725.
12. Heckbert, P. Color image quantization for frame buffer display. *ACM SIGGRAPH Comput. Graph*., 1982, **16**, (3), 297–307.
13. Gervautz, M. and Purgathofer, W. New Trends in Computer Graphics, 1988, Ch. A Simple Method for Color Quantization: Octree Quantization, pp. 219–231 (Springer, Berlin/New York).
14. Wan, S. J. and Wong, S. K. M. Prusinkiewicz, variance-based color image quantization for frame buffer display. *Color Res. Appl*., 1990, **15**, 52–58.
15. Orchard, M. and Bouman, C. Color quantization of images. *IEEE Trans. Signal Process*., 1991, **39**, 2677–2690.
16. Wu, X. Graphics Gems Volume II, 1991, Ch. Efficient Statistical Computations for Optimal Color Quantization, pp. 126–133 (Academic Press, San Diego, CA).
17. Joy, G. and Xiang, Z. Center-cut for color image quantization. *Visual Comput*., 1993, **10**, 62–66.
18. Yang, C.-Y. and Lin, J.-C. RWM-cut for color image quantization. *Comput. Graph*., 1996, **20**, (4), 577–588.
19. Hsieh, I.-S. and Fan, K.-C. An adaptive clustering algorithm for color quantization. *Pattern Recognit. Lett*., 2000, **21**, 337–346.
20. Cheng, S. and Yang, C. Fast and novel technique for color quantization using reduction of color space dimensionality. *Pattern Recognit. Lett*., 2001, **22**, (8), 845–856.
21. Lo, K. Chan, Y. and Yu, M. Colour quantization by three-dimensional frequency diffusion. *Pattern Recognit. Lett*., 2003, **24**, (14), 2325–2334.
22. Sirisathitkul, Y., Auwatanamongkol, S. and Uyyanonvara, B. Color image quantization using distances between adjacent colors along the color axis with highest color variance. *Pattern Recognit. Lett*., 2004, **25**, (9), 1025–1043.
23. Kanjanawanishkul, K. and Uyyanonvara, B. Novel fast color reduction algorithm for time-constrained applications. *J. Visual Commun. Image Represent*., 2005, **16**, (3), 311–332.
24. Equitz, W. H. A new vector quantization clustering algorithm. *IEEE Trans. Acoust., Speech Signal Process*., 1989, **37**, (10), 1568–1575.
25. Balasubramanian, R. and Allebach, J. A new approach to palette selection for color images. *J. Imag. Technol*., 1991, **17**, (6), 284–290.
26. Xiang, Z. and Joy, G. Color image quantization by agglomerative clustering. *IEEE Comput. Graph. Appl*., 1994, **14**, (3), 44–48.
27. Velho, L., Gomez, J. and Sobreiro, M. V. R. Color image quantization by pairwise clustering, Proc. 10th Brazilian Symp. on *Computer Graphics and Image Processing*, Campos do Jorddao, Brazil, October 1997, IEEE, pp. 203–210.
28. Brun, L. and Mokhtari, M. Two high speed color quantization algorithms, Proc. 1st Int. Conf. on *Color in Graphics and Image Processing*, Saint-Etienne, France, October 2000, pp. 116–121.
29. Goldberg, N. Colour image quantization for high resolution graphics display. *Image Vis. Comput*., 1991, **9**, (5), 303–312.
30. Xiang, Z. Color image quantization by minimizing the maximum intercluster distance. *ACM Trans. Graph*., 1997, **16**, (3), 260–276.
31. Kasuga, H., Yamamoto, H. and Okamoto, M. Color quantization using the fast k-means algorithm. *Syst. Comput. Jpn*, 2000, **31**, (8), 33–40.
32. Huang, Y.-L. and Chang, R.-F. A fast finite-state algorithm for generating RGB palettes of color quantized images. *J. Inf. Sci. Eng*., 2004, **20**, (4), 771–782.
33. Hu, Y.-C. and Lee, M.-G. K-means based color palette design scheme with the use of stable flags. *J. Electron. Imag*., 2007, **16**, (3), 033003.
34. Hu, Y.-C. and Su, B.-H. Accelerated k-means clustering algorithm for colour image quantization. *Imag. Sci. J*., 2008, **56**, (1), 29–40.
35. Celebi, M. E. Improving the performance of k-means for color quantization. *Image Vis. Comput*., 2011, **29**, (4), 260–271.
36. Frackiewicz, M. and Palus, H. Computational Vision and Medical Image Processing: Recent Trends, 2011, Ch. KM and KHM Clustering Techniques for Colour Image Quantisation, pp. 161–174 (Springer, Berlin).
37. Verevka, O. and Buchanan, J. Local k-means algorithm for colour image quantization, Proc. Graphics/Vision Interface Conf., Quebec Canada, May 1995, pp. 128–135.
38. Scheunders, P. Comparison of clustering algorithms applied to color image quantization. *Pattern Recognit. Lett*., 1997, **18**, (11–13), 1379–1384.
39. Cak, S., Dizdar, E. N. and Ersak, A. A fuzzy colour quantizer for renderers. *Displays*, 1998, **19**, (2), 61–65.
40. Ozdemir, D. and Akarun, L. Fuzzy algorithm for color quantization of images. *Pattern Recognit*., 2002, **35**, (8), 1785–1791.
41. Kim, D.-W., Lee, K. and Lee, D. A novel initialization scheme for the fuzzy c-means algorithm for color clustering. *Pattern Recognit. Lett*., 2004, **25**, (2), 227–237.
42. Schaefer, G. and Zhou, H. Fuzzy clustering for colour reduction in images. *Telecommun. Syst*., 2009, **40**, (1–2), 17–25.
43. Zhou, H., Schaefer, G., Sadka, A. and Celebi, M. E. Anisotropic mean shift based fuzzy c-means segmentation of dermoscopy images. *IEEE J. Sel. Top. Signal Process*., 2009, **3**, (1), 26–34.
44. Schaefer, G. and Zhou, H. Foundations of Computational Intelligence Volume 2, 2009, Ch. An Overview of Fuzzy C-Means Based Image Clustering Algorithms, pp. 295–310 (Springer, Berlin/Heidelberg).
45. Wen, Q. and Celebi, M. E. Hard vs. fuzzy c-means clustering for color quantization, *EURASIP J. Adv. Signal Process*., 2011, (1), 118–129.
46. Schaefer, G. Innovations in Intelligent Image Analysis, 2011, Ch. Intelligent Approaches to Colour Palette Design, pp. 275–289 (Springer, Berlin).
47. Bing, Z., Junyi, S. and Qinke, P. An adjustable algorithm for color quantization. *Pattern Recognit. Lett*., 2004, **25**, (16), 1787–1797.
48. Dekker, A. Kohonen neural networks for optimal colour quantization. *Netw.: Comput. Neural Syst*., 1994, **5**, (3), 351–367.
49. Papamarkos, N., Atsalakis, A. and Strouthopoulos, C. Adaptive color reduction. *IEEE Trans. Syst., Man, Cybern. Part B*, 2002, **32**, (1), 44–56.
50. Chang, C.-H., Xu, P., Xiao, R., and Srikanthan, T. New adaptive color quantization method based on self-organizing maps. *IEEE Trans. Neural Netw*., 2005, **16**, (1), 237–249.
51. Rasti, J., Monadjemi, A. and Vafaei, A. Color reduction using a multi-stage kohonen self-organizing map with redundant features. *Expert Syst. Appl*., 2011, **38**, (10), 13188–13197.
52. Chung, K.-L., Huang, Y.-H., Wang, J.-P. and Cheng, M.-S. Speedup of color palette indexing in self-organization of kohonen feature map. *Expert Syst. Appl*., 2012, **39**, (3), 2427–2432.
53. Xiao, Y., Leung, C.-S., Lam, P.-M. and Ho, T.-Y. Self-organizing map-based color palette for high-dynamic range texture compression. *Neural Comput. Appl*., 2012, **21**, (4), 639–647.
54. Uchiyama, T. and Arbib, M. An algorithm for competitive learning in clustering problems. *Pattern Recognit*., 1994, **27**, (10), 1415–1421.
55. Aloise, D., Deshpande, A., Hansen, P. and Popat, P. NP-hardness of euclidean sum-of-squares clustering. *Mach. Learn*., 2009, **75**, (2), 245–248.
56. Mahajan, M., Nimbhorkar, P. and Varadarajan, K. The planar k-means problem is NP-hard. *Theor. Comput. Sci*., 2012, **442**, 13–21.

57. Tarsitano, A. A computational study of several relocation methods for k-means algorithms. *Pattern Recognit*. 2003, **36**, (12), 2955–2966.
58. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inform. Theory*, 1982, **28**, (2), 129–136.
59. Celebi, M. E., Kingravi, H. and Vela, P. A. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Syst. Appl*., 2013, **40**, 200–210.
60. Xu, R. and Wunsch, D. Clustering, 2008 (Wiley-IEEE Press, New York).
61. Rumelhart, D. and Zipser, D. Feature discovery by competitive learning. *Cogn. Sci*., 1985, **9**, (1), 75–112.
62. Bratley, P. and Fox, B. L. Algorithm 659: implementing Sobol's quasirandom sequence generator. *ACM Trans. Math. Softw*., 1988, **14**, (1), 88–100.
63. Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. Numerical Recipes, 2007, 3rd edition, (Cambridge University Press, New York).
64. Matsumoto, M. and Nishimura, T. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul*., 1998, **8**, (1), 3–30.
65. Heath, M. T. Scientific Computing: An Introductory Survey, 2002, 2nd edition (McGraw-Hill, New York).
66. Braudaway, G. W. Procedure for optimum choice of a small number of colors from a large color palette for color imaging, Proc. Electronic Imaging Conf., San Francisco, CA February 1987, SPIE, pp. 71–75.
67. Ward, J. Hierarchical grouping to optimize an objective function. *J. Am. Stat. Assoc*., 1963, **58**, (301), 236–244.