EFFECTIVE INITIALIZATION OF K-MEANS FOR COLOR QUANTIZATION

M. Emre Celebi

Department of Computer Science
Louisiana State University, Shreveport, LA, USA
ecelebi@lsus.edu

ABSTRACT

Color quantization is an important operation with many applications in graphics and image processing. Most quantization methods are essentially based on data clustering algorithms. However, despite its popularity as a general purpose clustering algorithm, k-means has not received much respect in the color quantization literature because of its high computational requirements and sensitivity to initialization. In this paper, we investigate the performance of k-means as a color quantizer. We implement fast and exact variants of k-means with different initialization schemes and then compare the resulting quantizers to some of the most popular quantizers in the literature. Experiments on a set of classic test images demonstrate that an efficient implementation of k-means with an appropriate initialization strategy can in fact serve as a very effective color quantizer.

Index Terms— Color quantization, clustering, k-means, k-means++

1. INTRODUCTION

True-color images typically contain thousands of colors, which makes their display, storage, and processing problematic. For this reason, color quantization (reduction) is commonly used as a preprocessing step for various graphics and image processing tasks. In the past, color quantization was a necessity due to the limitations of the display hardware, which could not handle the 16 million possible colors in 24-bit images. Although 24-bit display hardware has become more common, color quantization still maintains its practical value [1]. Modern applications of color quantization include: (i) image compression, (ii) image segmentation, (iii) image analysis, and (iv) content-based image retrieval.

The process of color quantization is mainly comprised of two phases: palette design (the selection of a small set of colors that represents the original image colors) and pixel mapping (the assignment of each input pixel to one of the palette colors). The primary objective is to reduce the number of unique colors, 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 colors, faithful representation of these images with a limited size palette is a difficult problem.

Color quantization methods can be broadly classified into two categories: image-independent methods that determine a universal (fixed) palette without regard to any specific image, and image-dependent methods that determine a custom (adaptive) palette based on the color 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 quantization output.

Numerous image-dependent color quantization methods have been developed in the past three decades. These can be categorized into two families: preclustering methods and postclustering methods [1]. Preclustering methods are mostly based on the statistical analysis of the color distribution of the images. Divisive preclustering methods start with a single cluster that contains all N image pixels. This initial cluster is recursively subdivided until K clusters are obtained. Well-known divisive methods include median-cut [2], variance-based method [3], and greedy orthogonal bipartitioning [4]. On the other hand, agglomerative preclustering methods [5, 6] start with N singleton clusters each of which contains one image pixel. These clusters are repeatedly merged until K clusters remain. In contrast to preclustering methods that compute the palette only once, postclutering 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 optimization, they can obtain higher quality results when compared to preclustering methods at the expense of increased computational time. Clustering algorithms adapted to color quantization include k-means [7, 8], competitive learning [9, 10], fuzzy c-means [11], and self-organizing maps [12].

In this paper, we investigate the performance of the k-means clustering algorithm as a color quantizer. We implement four efficient k-means variants each one with a different

This publication was made possible by a grant from The Louisiana Board of Regents (LEQSF2008-11-RD-A-12).

initialization scheme and then compare these to some of the most popular color quantizers on a set of classic test images.

The rest of the paper is organized as follows. Section 2 describes the k-means clustering algorithm and various techniques for initializing the cluster centers. Section 3 presents the comparison of the k-means variants with other color quantization methods. Finally, Section 4 gives the conclusions.

2. COLOR QUANTIZATION USING K-MEANS CLUSTERING ALGORITHM

The k-means (KM) algorithm is inarguably one of the most widely used methods for data clustering. Given a data set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \in \mathbb{R}^D$, the objective of KM is to partition X into K exhaustive and mutually exclusive clusters $S = \{S_1, \ldots, S_k\}$ by minimizing the sum of squared error $\text{SSE} = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in S_k} \|\mathbf{x}_i - \mathbf{c}_k\|^2$ where $\|\|$ denotes the Euclidean (L_2) norm and $\mathbf{c}_k$ is the center of cluster S_k calculated as the mean (average) of points that belong to this cluster. This problem is known to be computationally intractable, but a heuristic method developed by Lloyd [13] offers a simple solution. Lloyd's algorithm starts with K arbitrary centers, typically chosen uniformly at random from the data points. Each point is then assigned to the nearest center, and each center 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 Algo. (1):

input : $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \in \mathbb{R}^D$ ($N \times D$ input data set)
output: $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\} \in \mathbb{R}^D$ (K cluster centers)
Select a random subset C of X as the initial set of cluster centers;
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;
    $m[i] = \underset{k \in \{1,\ldots,K\}}{\text{argmin}} \|\mathbf{x}_i - \mathbf{c}_k\|^2$;
  end
  Recalculate the cluster centers;
  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 center $\mathbf{c}_k$;
    $S_k = \{\mathbf{x}_i \,|m[i] = k\}$;
    Calculate the new center $\mathbf{c}_k$ 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

Algorithm 1: Conventional K-Means Algorithm

Here $m[i]$ denotes the membership of point $\mathbf{x}_i$, i.e. index of the cluster center that is nearest to $\mathbf{x}_i$.

When compared to the preclustering methods, there are two problems with using the KM algorithm for color quantization. First, due to its iterative nature, the algorithm might require an excessive amount of time to obtain an acceptable output quality. Second, the output is quite sensitive to the initial choice of the cluster centers.

The KM algorithm is computationally demanding mainly because the assignment phase of the algorithm involves many redundant distance calculations. In particular, for each point, the distances to each of the K cluster centers are calculated. Consider a point $\mathbf{x}_i$, two cluster centers $\mathbf{c}_a$ and $\mathbf{c}_b$ and a distance metric d, using the triangle inequality, we have $d(\mathbf{c}_a, \mathbf{c}_b) \leq d(\mathbf{x}_i, \mathbf{c}_a) + d(\mathbf{x}_i, \mathbf{c}_b)$. Therefore, if we know that $2d(\mathbf{x}_i, \mathbf{c}_a) \leq d(\mathbf{c}_a, \mathbf{c}_b)$, we can conclude that $d(\mathbf{x}_i, \mathbf{c}_a) \leq d(\mathbf{x}_i, \mathbf{c}_b)$ without having to calculate $d(\mathbf{x}_i, \mathbf{c}_b)$. The compare-means algorithm [14] precalculates the pairwise distances between cluster centers at the beginning of each iteration. When searching for the nearest cluster center for each point, the algorithm often avoids a large number of distance calculations with the help of the triangle inequality test. The sort-means (SM) algorithm [14] further reduces the number of distance calculations by sorting the distance values associated with each cluster center in ascending order. At each iteration, point $\mathbf{x}_i$ is compared against the cluster centers in increasing order of distance from the center $\mathbf{c}_k$ that $\mathbf{x}_i$ was assigned to in the previous iteration. If a center that is far enough from $\mathbf{c}_k$ is reached, all of the remaining centers can be skipped and the procedure continues with the next point. In this way, SM avoids the overhead of going through all the centers. Note that for data sets with low dimensionality such as color image data, SM is actually faster than the more elaborate algorithm proposed by Elkan [15] due to its low overhead.

Several initialization schemes have been proposed for KM:

- **Forgy** [16]: This is the simplest and most common initialization scheme, where the cluster centers are chosen randomly from the data set.
- **Minmax (Farthest First)** [17]: The first center $\mathbf{c}_1$ is chosen randomly and the i-th center $\mathbf{c}_i$ is chosen to be the point that has the largest minimum distance to the previously selected centers, i.e. $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{i-1}$.
- **Subset Farthest First** [18]: One drawback of the Minmax technique is that it tends to find the outliers in the data set. Using a smaller subset of size $2K \ln K$, the total number of outliers that Minmax can find is reduced and thus the proportion of nonoutlier points obtained as centers is increased.
- **K-Means++** [19]: The first center $\mathbf{c}_1$ is chosen randomly and the i-th center $\mathbf{c}_i$ is chosen to be $\mathbf{x}' \in X$ with a probability of $\frac{D(\mathbf{x}')^2}{\sum_{i=1}^{N} D(\mathbf{x}_i)^2}$, where $D(\mathbf{x})$ denote the minimum distance from a point $\mathbf{x}$ to the previously selected centers.

Among these Forgy's scheme is the simplest and most commonly used one. However, as will be seen in the next section, this scheme often leads to poor clustering results.

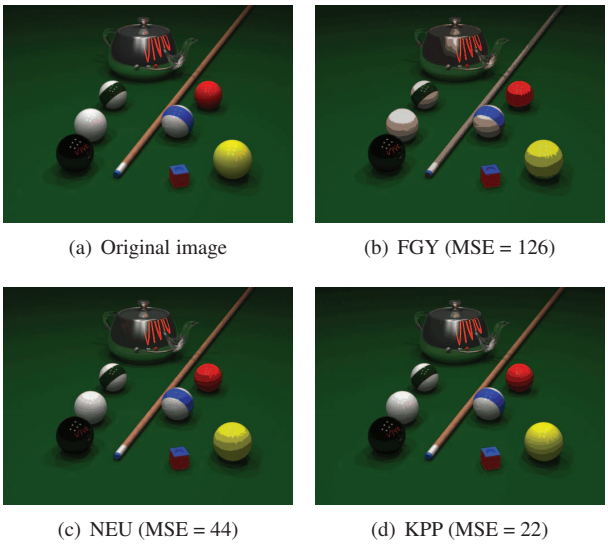
3. EXPERIMENTAL RESULTS AND DISCUSSION

The four variants of the KM algorithm (Forgy - FGY, Minmax - MMX, Subset Farthest First - SFF, and K-Means++ - KPP) were compared to some of the well-known quantization methods in the literature. These included median-cut (MC) [2], variance-based method (WAN) [3], greedy orthogonal bipartitioning (WU) [4], self-organizing map (NEU) [12] (with a sampling factor of 1, which is the highest quality configuration), and local k-means (LKM) [9]. The number of iterations for the KM variants was set to 10.

The effectiveness and efficiency of the quantization methods were quantified by Mean Squared Error (MSE) and CPU time (measured in milliseconds on an Intel®Core™2 Quad Q6700 2.66 GHz machine), respectively. The time figures were averaged over 50 runs. Table 1 shows the performance of the methods at quantization levels 64, 128, and 256 on some of the most commonly used test images in the literature. Three of these images (Baboon, Lenna, Peppers) are natural and one of them is synthetic (Poolballs). The best (lowest) MSE (E) values are shown in **bold**. Note that the initialization schemes presented in the previous section are all nondeterministic. Therefore, in the table, the quantization errors for these methods are specified in the form of mean/standard deviation over 50 runs. The following observations are in order:

- In general, the fastest method is MC, which is followed by WAN and WU. The slowest method is KPP.
- The time requirements for MC, WAN, and WU are more or less independent of K. In contrast, the requirements for the postclustering methods NEU, LKM, FGY, MMX, SFF, and KPP increase linearly with K.
- In general, the most effective method is KPP, which is followed by SFF, NEU, and WU. The least effective method is MC.
- The effectiveness of FGY and MMX fluctuates over the image set. On the natural images they perform quite well, whereas on the synthetic Poolballs image they obtain the high quantization errors.
- Despite the fact that all of the initialization schemes involve randomness, based on the standard deviation values, it can be concluded that SFF and KPP are the most consistent schemes, whereas FGY is the least consistent one. This was expected because FGY involves purely random initialization with no intelligent mechanism to avoid choosing non-representative centers.
- Overall, SFF achives the best balance between efficiency and effectiveness.

Figure 1 shows sample quantization results for the Poolballs image. It can be seen that FGY and NEU generate severe contouring especially on the red and yellow balls. In contrast, KPP obtains visually pleasing results with less prominent contouring.



(a) Original image (b) FGY (MSE = 126)

(c) NEU (MSE = 44) (d) KPP (MSE = 22)

Fig. 1. Sample quantization results ($K = 64$)

4. CONCLUSIONS

In this paper, the k-means clustering algorithm was investigated from a color quantization perspective. Several variants of k-means each one with a different initialization scheme were compared against commonly used color quantization methods. Experiments on a set of classic test images demonstrated that an efficient implementation of k-means with an appropriate initialization scheme can in fact serve as a very effective color quantizer.

5. REFERENCES

[1] L. Brun and A. Trémeau, *Digital Color Imaging Handbook*, chapter Color Quantization, pp. 589–638, CRC Press, 2002.

[2] P.S. Heckbert, "Color Image Quantization for Frame Buffer Display," *ACM SIGGRAPH Comp Graph*, vol. 16, no. 3, pp. 297–307, 1982.

[3] S.J. Wan, P. Prusinkiewicz, and S.K.M. Wong, "Variance-Based Color Image Quantization for Frame Buffer Display," *Col Res Appl*, vol. 15, no. 1, pp. 52–58, 1990.

[4] X. Wu, *Graphics Gems Volume II*, chapter Efficient Statistical Computations for Optimal Color Quantization, pp. 126–133, Academic Press, 1991.

Table 1. Comparison of color quantization methods (K: # colors, T: CPU time in milliseconds, E: MSE)

	Baboon (512×512 pixels, 153,171 colors)					
	K = 64		K = 128		K = 256	
Method	T	E	T	E	T	E
MC	20	371	22	248	23	166
WAN	27	326	29	216	34	142
WU	30	248	30	155	31	99
NEU	135	216	261	128	489	84
LKM	250	216	429	145	770	92
FGY	233	206/5	361	129/2	648	83/1
MMX	363	204/2	609	130/1	1162	86/0
SFF	231	202/2	353	127/1	622	82/0
KPP	946	**199**/1	1782	**125**/0	3469	**79**/0
	Lenna (512×480 pixels, 56,164 colors)					
MC	19	94	18	71	22	47
WAN	24	93	27	61	34	43
WU	29	76	28	46	31	29
NEU	124	68	249	36	481	23
LKM	239	73	351	42	604	33
FGY	204	61/2	321	38/1	607	24/1
MMX	324	59/1	561	37/0	1077	24/0
SFF	206	58/1	320	36/0	600	23/0
KPP	862	**57**/0	1654	**35**/0	3242	**22**/0
	Peppers (512×512 pixels, 111,344 colors)					
MC	21	213	22	147	21	98
WAN	28	215	28	142	32	93
WU	30	160	30	101	31	63
NEU	136	151	263	83	495	55
LKM	251	173	427	100	633	62
FGY	222	141/4	348	86/ 2	630	54/1
MMX	354	141/2	603	88/1	1141	56/0
SFF	225	136/1	346	84/1	623	53/0
KPP	916	**133**/1	1758	**81**/0	3415	**50**/0
	Poolballs (510×383 pixels, 13,604 colors)					
MC	16	64	17	38	18	27
WAN	20	59	23	45	25	38
WU	22	31	22	17	24	11
NEU	106	44	196	18	354	9
LKM	161	69	265	30	473	18
FGY	144	126/21	214	78/22	394	38/5
MMX	173	71/13	297	35/7	619	16/2
SFF	97	28/3	143	13/2	308	6/0
KPP	593	**22**/0	1138	**10**/0	2301	**5**/0

[5] R. Balasubramanian and J.P. Allebach, "A New Approach to Palette Selection for Color Images," *J Imag Tech*, vol. 17, no. 6, pp. 284–290, 1991.

[6] L. Velho, J. Gomez, and M.V.R. Sobreiro, "Color Image Quantization by Pairwise Clustering," in *Proc of SIBGRAPI'97*, 1997, pp. 203–210.

[7] Y.-C. Hu and M.-G. Lee, "K-means Based Color Palette Design Scheme with the Use of Stable Flags," *J Electron Imag*, vol. 16, no. 3, pp. 033003, 2007.

[8] Y.-C. Hu and B.-H. Su, "Accelerated K-means Clustering Algorithm for Colour Image Quantization," *Imag Sci J*, vol. 56, no. 1, pp. 29–40, 2008.

[9] O. Verevka and J.W. Buchanan, "Local K-Means Algorithm for Colour Image Quantization," in *Proc of GI'95*, 1995, pp. 128–135.

[10] M.E. Celebi, "An Effective Color Quantization Method Based on the Competitive Learning Paradigm," in *Proc of IPCV'09*, 2009, vol. 2, pp. 876–880.

[11] D. Ozdemir and L. Akarun, "Fuzzy Algorithm for Color Quantization of Images," *Pattern Recogn*, vol. 35, no. 8, pp. 1785–1791, 2002.

[12] A.H. Dekker, "Kohonen Neural Networks for Optimal Colour Quantization," *Netw Comput Neural Syst*, vol. 5, no. 3, pp. 351–367, 1994.

[13] S.P. Lloyd, "Least Squares Quantization in PCM," *IEEE Trans Inform Theor*, vol. 28, no. 2, pp. 129–136, 1982.

[14] S.J. Phillips, "Acceleration of K-Means and Related Clustering Algorithms," in *Proc of ALENEX'02*, 2002, pp. 166–177.

[15] C. Elkan, "Using the Triangle Inequality to Accelerate k-Means," in *Proc of ICML'03*, 2003, pp. 147–153.

[16] E. Forgy, "Cluster Analysis of Multivariate Data: Efficiency vs. Interpretability of Classification," *Biometrics*, vol. 21, pp. 768, 1965.

[17] D. Hochbaum and D. Shmoys, "A Best Possible Heuristic for the k-Center Problem," *Math Oper Res*, vol. 10, no. 2, pp. 180–184, 1985.

[18] D. Turnbull and C. Elkan, "Fast Recognition of Musical Genres Using RBF Networks," *IEEE Trans Knowl Data Eng*, vol. 2005, no. 4, pp. 580–584, 17.

[19] D. Arthur and S. Vassilvitskii, "k-means++: The Advantages of Careful Seeding," in *Proc of SODA'07*, 2007, pp. 1027–1035.