2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
July 6-11, 2014, Beijing, China

Fast Color Reduction Using Approximative c-Means Clustering Models

László Szilágyi[1], Gellért Dénesi[2] and Sándor M. Szilágyi[3]

***Abstract*— In this paper we propose an efficient color reduction framework that employs c-means clustering to extract optimal colors. The processing consists of three stages: preprocessing, c-means clustering, and creation of the output image. The main goal of the first stage is to transform the pixel matrix into a list of records, which indicates what colors are present in the image and how many times they appear. To achieve this, first we apply a static color quantization scheme that aligns the 16.7 million possible colors with 140 thousand grid points, and build the histogram of this quantized image. Then we mark least frequent quantized colors to be ignored during the clustering stage, the amount of such marks being controlled by the pixel inclusion parameter. Leaving out $2-5\%$ of the image pixels can reduce the number of colors to 500-5000 in most images. This limited set of colors together with frequency information consists the input of the c-means clustering process performed in the second stage. Before creating the final output image, the marked quantized colors are mapped to the closest cluster. Thorough numerical tests were performed on 500 randomly chosen images using both fuzzy and hard c-means clustering. Evaluations revealed that hard c-means is more suitable than fuzzy c-means for the given problem, both in terms of accuracy and efficiency. The proposed method performs quicker 2-3 times than other recent reported solutions.**

I. INTRODUCTION

Data reduction has always been widely researched due to limitations in storage space and communication bandwidth [4]. The relevant issue in most such applications is to retain the meaning of the data as much as possible while reducing its size. This requirement led to several optimal methods which employed self-organizing maps [15], self growing and self organized neural gas [2], ant colony [8], Fibonacci lattices [12], superposed histogram based adaptive clustering [9], dynamic programming [10], modified fuzzy c-means clustering [14], and improved hard c-means clustering [5]. Besides effective storage and communication, the most relevant applications of color reduction are in image segmentation [6] and document analysis [13].

Research supported by the Hungarian National Research Funds (OTKA), Project no. PD103921, and the Hungarian Academy of Sciences through the János Bolyai Fellowship program.

[1]L. Szilágyi is with Dept. of Control Engineering and Information Technology, Budapest University of Technology and Economics, Magyar tudósok krt. 1, 1117 Budapest, Hungary (phone: +36-1-463-4027; fax: +36-1-463-2699) and with Dept. of Electrical Engineering, Sapientia University, Calea Sighişoarei 1/C, 540485 Tîrgu Mureş, Romania (phone: +40-265-206-210; fax: +40-265-206-211; e-mail: lalo at ms.sapientia.ro).

[2]G. Dénesi is with Dept. of Electrical Engineering, Sapientia University, Calea Sighişoarei 1/C, 540485 Tîrgu Mureş, Romania (phone: +40-265-206-210; fax: +40-265-206-211; e-mail: denesigellert at yahoo.com).

[3]S. M. Szilágyi is with Dept. of Informatics, Petru Maior Univesity, Str. N. Iorga nr. 1, 540088 Tîrgu Mureş, Romania (phone/fax: +40-265-262-275; e-mail: szsandor72 at yahoo.com).

In this paper we extend the efficient histogram based fuzzy c-means clustering of single-channel images [17] to the case of color images. A static color quantization is first employed, after which the colors present in the image are listed together with their frequency. A previously defined parameter controls the amount of pixels to be ignored by the later processing. Quantized colors with low frequency value are excluded, while those above the extracted threshold are selected for the c-means clustering process. Both fuzzy and hard c-means clustering models are tested, on a large image set and a wide range of scenarios created by algorithm parameters, to establish which is the most accurate and efficient solution.

II. BACKGROUND

A. The fuzzy and hard c-means algorithms

The conventional FCM algorithm partitions a set of object data into a number of c clusters based on the minimization of a quadratic objective function. The objective function to be minimized is defined as:

$$J_{\mathrm{FCM}} = \sum_{i=1}^{c}\sum_{k=1}^{n} u_{ik}^{m} ||\mathbf{x}_k - \mathbf{v}_i||_A^2 = \sum_{i=1}^{c}\sum_{k=1}^{n} u_{ik}^{m} d_{ik}^2 \ , \quad (1)$$

where $\mathbf{x}_k$ represents the input data ($k = 1 \dots n$), $\mathbf{v}_i$ represents the prototype or centroid value or representative element of cluster i ($i = 1 \dots c$), $u_{ik} \in [0,1]$ is the fuzzy membership function showing the degree to which vector $\mathbf{x}_k$ belongs to cluster i, $m > 1$ is the fuzzyfication parameter, and d_{ik} represents the distance (any inner product norm defined by a symmetrical positive definite matrix A) between vector $\mathbf{x}_k$ and cluster prototype $\mathbf{v}_i$. FCM uses a probabilistic partition, meaning that the fuzzy memberships of any input vector $\mathbf{x}_k$ with respect to classes satisfy the probability constraint $\sum_{i=1}^{c} u_{ik} = 1$.

The minimization of the objective function J_{FCM} is achieved by alternately applying the optimization of J_{FCM} over $\{u_{ik}\}$ with $\mathbf{v}_i$ fixed, $i = 1 \dots c$, and the optimization of J_{FCM} over $\{\mathbf{v}_i\}$ with u_{ik} fixed, $i = 1 \dots c$, $k = 1 \dots n$ [3]. During each iteration, the optimal values are deduced from the zero gradient conditions and Lagrange multipliers, and obtained as follows:

$$u_{ik}^{\star} = \frac{d_{ik}^{-2/(m-1)}}{\sum_{j=1}^{c} d_{jk}^{-2/(m-1)}} \quad \forall\, i = 1 \dots c, \forall\, k = 1 \dots n \ , \quad (2)$$

$$\mathbf{v}_i^{\star} = \frac{\sum_{k=1}^{n} u_{ik}^{m} \mathbf{x}_k}{\sum_{k=1}^{n} u_{ik}^{m}} \quad \forall\, i = 1 \dots c \ . \quad (3)$$

978-1-4799-2072-3/14/$31.00 ©2014 IEEE

According to the alternating optimization scheme of the FCM algorithm, Eqs. (2) and (3) are alternately applied, until cluster prototypes stabilize. This stopping criterion compares the sum of norms of the variations of the prototype vectors $\mathbf{v}_i$ within the latest iteration, with a predefined small threshold value ε.

Hard c-means [16] is a special case of FCM, which uses $m = 1$, and thus the memberships are obtained by the winner-takes-all rule. Each cluster prototype will be the average of the input vectors assigned to the given cluster.

B. Histogram based quick c-means clustering

Both hard and fuzzy c-means clustering apply global optimization. In image processing this means none of them takes into account the position of pixels, as they are labeled according to their colors only. This property makes c-means clustering models suitable to histogram based efficient implementation, in which colors are clustered instead of pixels. In most single channel gray intensity images the number of gray levels is less than the number of pixels by several orders of magnitude, allowing for drastic reduction of processing time [17]. This paper will show how that previous solution can be extended to color images.

III. MATERIALS AND METHODS

The proposed method involves three main stages performed sequentially. Clustering millions of pixels with FCM or HCM does not fit in our target time frame. That is why in the preprocessing step we first aggregate pixels of similar color using the histogram and then select only frequent colors as input for the clustering. This way the HCM or FCM-based clustering performed in the second step will only need to group at most a few thousand vectors. The final phase of labels all colors present in the histogram and generates the output image by providing a label to each pixel. Details for each stage are described in the following sections.

A. Data aggregation and color selection

A color image has three data channels independent of each other. Since several color coding schemes exist, for the sake of generality we will denote image channels by α, β, and γ. Each pixel is described by a three-element vector of integers, having values between 0 and 255. This resolution allows for more than 16 millions of possible colors, this number exceeding the amount of pixels in the image. This resolution is capable to address a lot more colors than the human eye can distinguish, so we may reduce the number of possible intensity levels in each channel. Our proposed method rounds each intensity level to the closest integer number divisible by 5, cutting the number of possible colors down to $52^3 = 140,608$. The grid size of this color aggregation scheme was chosen empirically and was considered large enough to reduce the colors present in the image to the order of tens of thousands, and small enough to preserve the most important details of the image.

In the next step, we create a table of the colors that are present in the image, and denote it by $colBuffer$. Row i in this table describes the color with index i, and it contains the following variables:

1) three 8-bit unsigned integers α_i, β_i, and γ_i to store the three components of the given color (according to the employed image coding scheme);
2) a 32-bit unsigned integer ν_i initialized with the number of pixels having the given color;
3) flag σ_i which will show whether color i is included in the clustering process;
4) an 8-bit unsigned integer λ_i that indicates the labeling of color placed in row i.

The number of records in $colBuffer$ is stored in $nrCol$. In the following, the records of $colBuffer$ are referred to as aggregated colors.

The fast execution of the color reduction process is assisted by two auxiliary tables:

Data: image $f(x, y)$ with $1 \le x \le f.width$, $1 \le y \le f.height$; ratio $P \in [0, 1]$ of colors to keep
Result: $colBuffer = \{(\alpha_k, \beta_k, \gamma_k, \nu_k, \sigma_k, \lambda_k), 1 \le k \le nrCol\}$

```
hist2col ← zeros(52, 52, 52);
pix2col ← zeros(f.width, f.height);
k ← 1;
for each pixel (x, y) do
    [α, β, γ] ← round(f(x, y)/5);
    if hist2col(α, β, γ) = 0 then
        hist2col(α, β, γ) ← k;
        α_k ← α; β_k ← β; γ_k ← γ;
        ν_k ← 1;
        k ← k + 1;
    end
    else
        ν_k ← ν_k + 1;
    end
    pix2col(x, y) ← hist2col(α, β, γ);
end
free(hist2col);
nrCol ← k;
colHist ← zeros(...);
for k = 1 ... nrCol do
    colHist(ν_k) ← colHist(ν_k) + 1;
end
nrPix ← f.width × f.height;
keptPix ← nrPix;
θ ← 0;
while keptPix > P × nrPix do
    θ ← θ + 1;
    keptPix ← keptPix − theta × colHist(θ);
end
for k = 1 ... nrCol do
    σ_k ← (ν_k ≥ θ);
end
free(colHist);
```

Algorithm 1: Preprocessing for fast color reduction

1) $hist2col$ is a three-dimensional array of size 52 in each dimension, $hist2col(\alpha, \beta, \gamma)$ indicating the row index of color (α, β, γ) in the $colBuffer$. This table is needed only as long as $colBuffer$ is established.
2) $pix2col$ is a two-dimensional array having its size equal to the input image, $pix2col(x, y)$ indicating the row index of the color of pixel (x, y) in the $colBuffer$. This table will be used to speed up the generation of the final result (output image).

Data: color buffer $colBuffer$ with the exception of labels λ_k
Result: labels $\lambda_k, 1 \leq k \leq nrCol$; cluster prototypes $\mathbf{v}_i$, $i = 1 \dots c$
`Initialize` $\mathbf{v}_i$, $i = 1 \dots c$;
repeat
  $\Sigma_{\mathbf{v}} \leftarrow zeros(c, 3)$;
  $\Omega_{\mathbf{v}} \leftarrow zeros(c, 1)$;
  $\Phi \leftarrow 0$;
  for $k = 1 \dots nrCol$ **do**
    if $\sigma_k = true$ **then**
      $\lambda = \arg\min_i \{dist(\mathbf{v}_i, [\alpha_k, \beta_k, \gamma_k])\}$;
      $\Sigma_{\mathbf{v}}(\lambda) \leftarrow \Sigma_{\mathbf{v}}(\lambda) + \nu_k[\alpha_k, \beta_k, \gamma_k]$;
      $\Omega_{\mathbf{v}}(\lambda) \leftarrow \Omega_{\mathbf{v}}(\lambda) + \nu_k$;
      if $\lambda \neq \lambda_k$ **then**
        $\lambda_k \leftarrow \lambda$;
        $\Phi \leftarrow \Phi + 1$;
      end
    end
  end
  for $i = 1 \dots c$ **do**
    $\mathbf{v}_i \leftarrow \Sigma_{\mathbf{v}}(i)/\Omega_{\mathbf{v}}(i)$;
  end
until $\Phi = 0$;
$free(\Sigma_{\mathbf{v}})$;
$free(\Omega_{\mathbf{v}})$;

Algorithm 2: Hard c-means implementation

The preprocessing algorithm is presented in details in Algorithm 1. At the point when $colBuffer$ is established, table $hist2col$ is released, and the color selection process begins, which excludes those aggregated colors from the later clustering, which are represented by few pixels. This color elimination is controlled by input parameter P, which defines the ratio of pixels to be kept for clustering. A histogram of aggregated colors is created ($colHist$): $colHist(q)$ indicates the number of aggregated colors to which exactly q pixels belong. This histogram helps us find the threshold θ that stores the minimum number of pixels an aggregated color needs to have in order to be kept for the clustering. The last preprocessing task sets the selection labels σ_k, for each aggregated color $k = 1 \dots nrCol$.

B. Hard or fuzzy clustering of aggregated colors

The c-means clustering is employed to group those aggregated colors, which were selected in the previous stage,

Data: color buffer $colBuffer$ with the exception of labels λ_k
Result: labels $\lambda_k, 1 \leq k \leq nrCol$; cluster prototypes $\mathbf{v}_i$, $i = 1 \dots c$
`Initialize` $\mathbf{v}_i$, $i = 1 \dots c$;
for $i = 1 \dots c$ **do**
  $\mathbf{v}_i^{(old)} \leftarrow \mathbf{v}_i$;
end
repeat
  $\Sigma_{\mathbf{v}} \leftarrow zeros(c, 3)$;
  $\Omega_{\mathbf{v}} \leftarrow zeros(c, 1)$;
  for $k = 1 \dots nrCol$ **do**
    if $\sigma_k = true$ **then**
      for $i = 1 \dots c$ **do**
        $d_i \leftarrow dist(\mathbf{v}_i, [\alpha_k, \beta_k, \gamma_k])$;
      end
      $q \leftarrow \arg\min_i \{d_i, i = 1 \dots c)\}$;
      if $d_q = 0$ **then**
        for $i = 1 \dots c$ **do**
          $u_i \leftarrow 0$;
        end
        $u_q \leftarrow 1$;
      end
      else
        $\Sigma_u \leftarrow 0$;
        for $i = 1 \dots c$ **do**
          $u_i \leftarrow d_i^{-2/(m-1)}$;
          $\Sigma_u \leftarrow \Sigma_u + u_i$;
        end
        for $i = 1 \dots c$ **do**
          $u_i \leftarrow u_i/\Sigma_u$;
        end
      end
      for $i = 1 \dots c$ **do**
        $\Sigma_{\mathbf{v}}(i) \leftarrow \Sigma_{\mathbf{v}}(i) + \nu_k u_i^m[\alpha_k, \beta_k, \gamma_k]$;
        $\Omega_{\mathbf{v}}(i) \leftarrow \Omega_{\mathbf{v}}(i) + \nu_k u_i^m$;
      end
    end
  end
  $\Phi \leftarrow 0$;
  for $i = 1 \dots c$ **do**
    $\mathbf{v}_i \leftarrow \Sigma_{\mathbf{v}}(i)/\Omega_{\mathbf{v}}(i)$;
    $\Phi \leftarrow \Phi + ||\mathbf{v}_i - \mathbf{v}_i^{(old)}||$;
    $\mathbf{v}_i^{(old)} \leftarrow \mathbf{v}_i$;
  end
until $\Phi < \varepsilon$;
for $k = 1 \dots nrCol$ **do**
  if $\sigma_k = true$ **then**
    $\lambda_k \leftarrow \arg\min_i \{dist(\mathbf{v}_i, [\alpha_k, \beta_k, \gamma_k])\}$;
  end
end
$free(\Sigma_{\mathbf{v}})$;
$free(\Omega_{\mathbf{v}})$;

Algorithm 3: Fuzzy c-means implementation

having their σ_k flag set to $true$. Implementation details are presented in Algorithm 2 for HCM and Algorithm 3 for FCM. As proposed by Kolen and Hutcheson [11], partition information is not stored during the iterations. Instead of that, each cluster with index i ($i = 1 \dots c$) is associated with a weighted sum of input color vectors assigned to it $\Sigma_{\mathbf{v}}(i)$ and the number of input pixels assigned to the cluster $\Omega_{\mathbf{v}}(i)$. At the end of each loop, the prototype of cluster i is updated with the ratio of $\Sigma_{\mathbf{v}}(i)$ and $\Omega_{\mathbf{v}}(i)$. FCM needs a c-element vector u to temporarily store fuzzy memberships of the currently handled aggregated color, while variable Σ_u supports the normalization of fuzzy memberships.

During each iteration, variable Φ counts the number of labels that change. The algorithm stops when the sum of changes Φ falls below a previously set threshold value ε. In case of HCM, the stopping condition may be set to $\Phi = 0$.

At the end of the clustering process, we obtain final prototypes for each cluster, and a label for each aggregated color that was selected for clustering.

C. Creating the output image

No matter whether hard or fuzzy clustering was performed, the output image is generated using the labels given to selected aggregated colors, and final cluster prototypes. The algorithm is summarized in Algorithm 4. First a label is associated to all aggregated colors corresponding to the closest cluster prototype. Finally the new color for each pixel is extracted from the cluster prototypes using the previously established $pix2col$ table.

Data: color buffer $colBuffer$; lookup table $pix2col$
Result: image $f(x, y)$ with $1 \le x \le f.width$, $1 \le y \le f.height$
for $k = 1 \dots nrCol$ **do**
  if $\sigma_k = false$ **then**
    $\lambda_k \leftarrow \arg\min_i \{dist(\mathbf{v}_i, (\alpha_k, \beta_k, \gamma_k))\}$;
  end
end
for *each pixel* (x, y) **do**
  $\lambda \leftarrow \lambda_{pix2col(x,y)}$;
  $f(x, y) \leftarrow \mathbf{v}_\lambda$;
end
$free(pix2col)$;

Algorithm 4: Postprocessing for fast color reduction

IV. RESULTS

A. Preprocessing large sets of images

The proposed algorithms have been tested using a randomly chosen set of 500 colored photographs, each created at the resolution of 10Mpixels. Before being fed to color reduction, various resized versions were generated for each image, thus test images ranged from 800×600 to 3648×2736 pixels. The number of aggregated colors were recorded for all 500 images. Images were sorted according to the increasing number of aggregated colors. Figure 1 reflects the number of aggregated colors present in these 500 images at the end of preprocessing, and selected colors in case of various values of parameter P. Table I shows us the percentage of selected colors for various values of parameter P, in case of certain typical images. Table I clearly indicates that we can reduce the amount of clustered data 2-3 times and ignore only 2% of the pixels; or making the clustered data set smaller by an order of magnitude is achievable through ignoring 15% of the pixels.

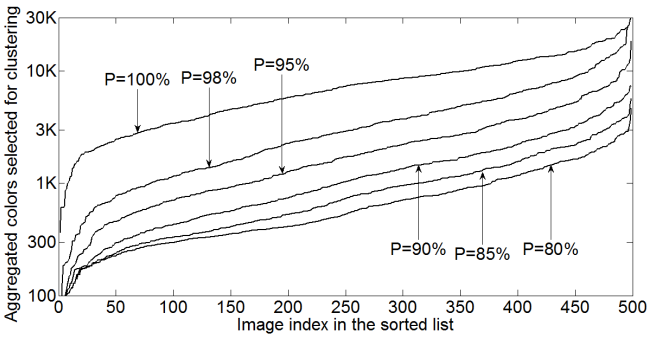

Fig. 1. Total runtime vs. aggregated color number, for various number of clusters

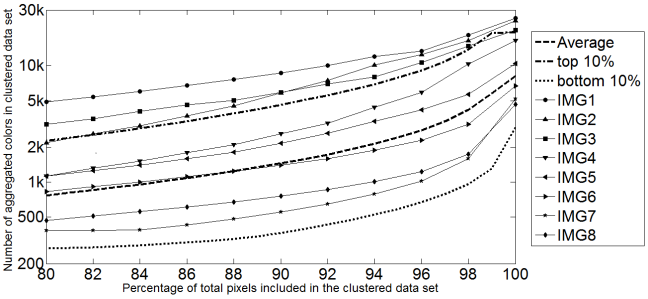

Fig. 2. Number of selected aggregated colors plotted vs. percentage of pixels included in the clustered data set (P), for various selected images

TABLE I
EFFECT OF PARAMETER P (PIXEL SELECTION RATIO) ON THE NUMBER OF AGGREGATED COLORS SELECTED FOR CLUSTERING, IN CASE OF TYPICAL IMAGES WITH LOW, AVERAGE, MEDIAN, AND HIGH AMOUNT OF AGGREGATED COLORS

Aggregated color amount	Pixel selection ratio (P)				
	98%	95%	90%	85%	80%
Low	33.81%	20.62%	12.60%	9.68%	8.66%
Average	39.59%	23.26%	13.99%	9.99%	7.21%
Median	40.24%	23.35%	14.15%	10.28%	8.78%
High	52.12%	29.86%	17.94%	12.59%	9.42%

From the initial large set of images we have selected eight, and denoted them by IMG1 to IMG8, indexed in the decreasing order of aggregated colors present in the images. IMG1-IMG3 are colorful images, while IMG6 is the closest to the image with median number of aggregated colors. Low-quality, less colorful, blurred images are not present in this

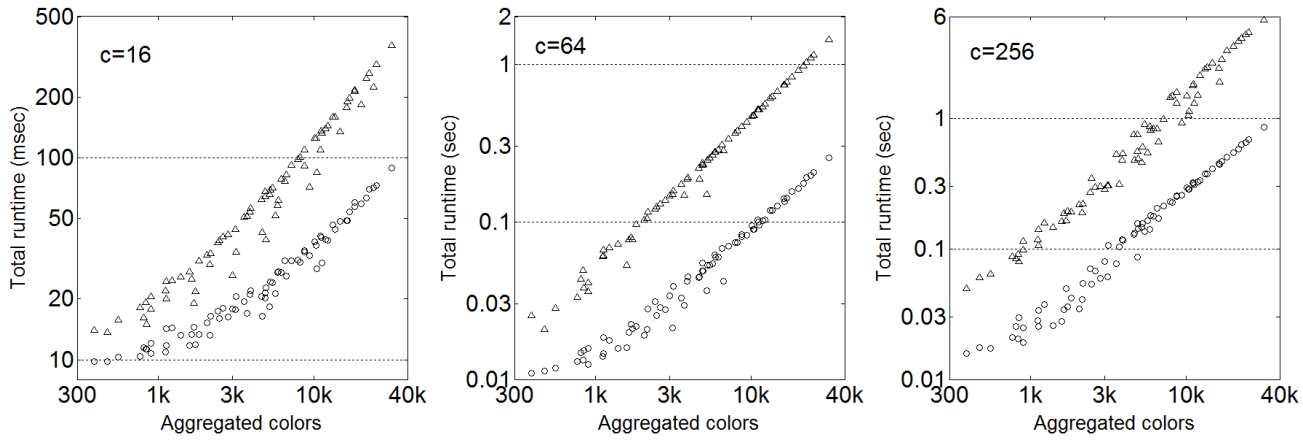

Fig. 3. Total runtime vs. number of selected aggregated colors, disregarding the input image and the percentage of pixels covered. Each plot represents the case of a different number of clusters. Triangles and circles indicate FCM and HCM runtimes, respectively.

reduced image set. These eight images will serve for detailed benchmarks of the color reduction algorithm.

Figure 2 indicates the amount of selected aggregated colors that cover P ratio of pixels, plotted against ratio P, in case of images IMG1-IMG8. It also exhibits two averaged curves computed from the top 10% and bottom 10% of the large set of images, and the averaged curve of the whole large image set. Here we can see that the number of selected aggregated colors necessary to cover a certain percentage of image pixels may differ from image to image by order of magnitude.

B. *Efficiency benchmarks*

The reduced set of images (IMG1-IMG8) underwent the efficiency benchmarking process, involving thousands of runs according to the following scheme:

- Image size varied in six steps from 800×600 to 3648×2736 pixels.
- The ratio P of included pixels had six different values between 80% to 100%.
- For the number of clusters c, eleven different values were chosen between 8 and 256.
- Both HCM and FCM were tested for each image and each setting. FCM always used fuzzy exponent $m = 2$.
- For each setting we executed seven different runs, recorded and sorted the total runtime for each run, and extracted the average of the middle three values.

All efficiency tests were run on a single core of a quad-core i7 processor running at 3.4GHz frequency. Figures reflect benchmark tests on 800×600 pixel sized images, unless otherwise stated.

Figures 3 and 4 report runtime values and make a comparison between HCM and FCM. Figure 3 plots runtimes against the number of selected aggregated colors included in the clustering process (number of rows in the $colBuffer$ for which selection flag σ_k is set true). Triangles indicate FCM runtimes, while circles stand for HCM ones. These plots do not indicate which was the input image or what percentage of the pixels was included in the clustering process. They only attempt to capture the trend between total runtime and the cardinality of the clustered data set. Apparently FCM represents 2-5 times higher computational burden than HCM, and the ratio is slightly growing if the number of clustered colors rises. The overall runtime visibly grows with the number of classes as well.

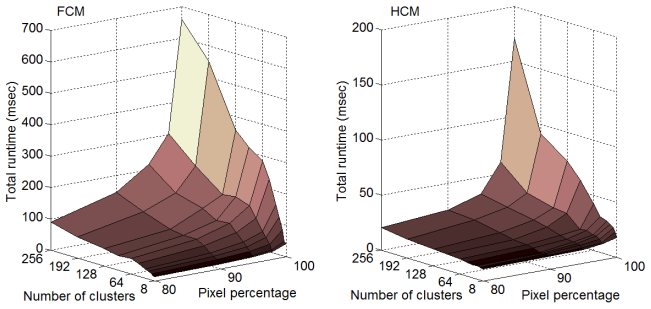

Fig. 4. Total runtime vs. percentage of included pixels P and cluster number c, in case of a typical image: FCM and HCM benchmarks on the left and right side, respectively

Figure 4 exhibits overall runtime values of FCM and HCM obtained on a typical image (IMG6), for various cluster numbers c and selected values of the included pixel percentage parameter P. It is clearly visible that the exclusion of a couple percent of the pixels significantly reduces the computational load of both algorithms. Again, we can see that HCM is much quicker than FCM. Considering the fact that the quality of FCM's output images is not at all better than the outcome of HCM which was also reported by Celebi et al [5], in the further efficiency benchmarking we will only show HCM runtimes.

Figure 5 shows us how the total runtime of the proposed color reduction algorithm depends on the number of clusters. A fixed ratio of $P = 95\%$ of the pixels were included into the clustering. The computational load apparently has a linear growth with the number of clusters. Figure 6 exhibits the evolution of HCM's total runtime when the percentage of selected pixels varies.

TABLE II

TOTAL RUNTIME (IN $msec$) OF THE HCM BASED ALGORITHM, IN CASE OF VARIOUS CLUSTER NUMBERS c AND PIXEL INCLUSION RATIO P, FOR THE COLORFUL IMAGE IMG2 AND TYPICAL IMAGE IMG6, AT VARIOUS SIZES OF THE INPUT IMAGES

IMG2 at resolution	800×600			2048×1536			3648×2736		
Pixel inclusion ratio P	100%	95%	80%	100%	95%	80%	100%	95%	80%
$c = 8$	33	21	16	130	105	101	329	296	292
$c = 16$	71	38	17	187	124	103	397	316	298
$c = 32$	115	58	21	256	147	106	479	339	307
$c = 64$	197	95	28	387	186	123	638	379	317
$c = 128$	353	161	32	638	263	127	940	455	338
$c = 256$	654	293	54	1118	407	153	1525	606	369
IMG6 at resolution	800×600			2048×1536			3648×2736		
Pixel inclusion ratio P	100%	95%	80%	100%	95%	80%	100%	95%	80%
$c = 8$	18	12	11	97	92	91	294	281	280
$c = 16$	26	15	11	106	93	92	302	285	283
$c = 32$	30	17	12	127	95	94	320	287	286
$c = 64$	42	19	13	143	99	95	349	293	291
$c = 128$	74	26	16	202	104	101	437	301	304
$c = 256$	171	35	21	333	114	109	601	321	313

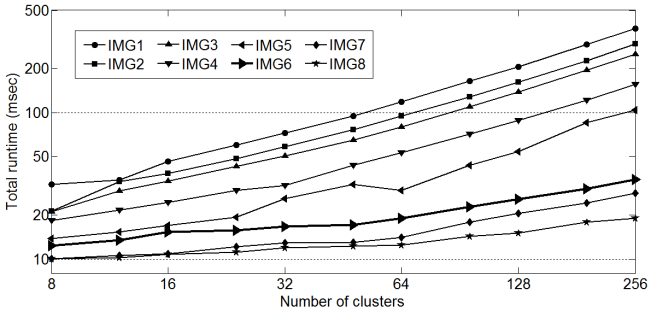

Fig. 5. Total runtime of HCM vs. number of clusters, at fixed $P = 95\%$, in case of the 8 selected images

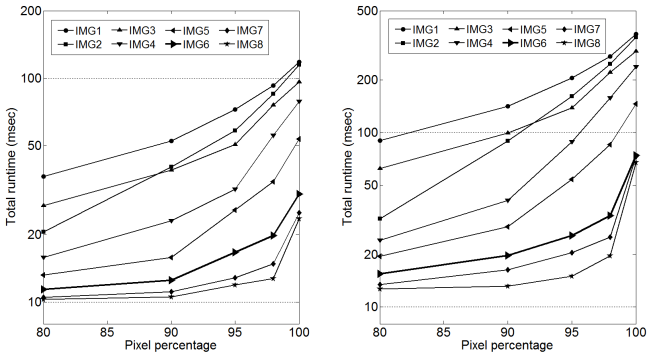

Fig. 6. Total runtime of HCM vs. pixel inclusion rate, at fixed number of clusters $c = 32$ (left) and $c = 128$ (right), in case of the 8 selected images

The total runtime linearly grows with the number of colors, when the images have low resolution. On the other hand, excluding no more than 5 percent of the pixels from the clustering process can reduce the total runtime up to 5 times.

Table II shows us how the size of the input image influences the total runtime in various circumstances. The runtime is not growing proportionally with the number of pixels, especially when the number of clusters is high.

C. Accuracy tests

Qualitative testing the accuracy is possible via visual inspection of output images. Figure 7 exhibits the outcome of the HCM based algorithm in case of six selected images. All these tests were run using pixel inclusion rate $P = 95\%$, while the number of clusters varied in range of 32-96. One has to look at minor details to recognize the difference between the input and output images. On the other hand, the FCM based algorithm distorts the colors in the output image, which can be explained with the fact that all pixels of the image have nonzero fuzzy memberships with respect to each cluster, thus contributing to the all cluster prototypes and producing clusters that have the prototypes of HCM shifted towards the grand mean (average color of the image).

To provide quantitative evaluation of the algorithm, we computed the averaged color distance (ACD) between the original input images and the final color reduced images in various scenarios, employing the formula:

$$\mathrm{ACD} = \frac{1}{n} \sum_{k=1}^{n} ||\mathbf{x}_k - \mathbf{v}_{\lambda_{pix2col(x,y)}}|| \ , \qquad (4)$$

where $\mathbf{x_k}$ is the color vector of pixel at position (x, y) and all other variables are those introduced in Section III. Figure 8 exhibits a logarithmic plot of the variation of ACT with respect to the cluster number c, for the typical image IMG6 and the colorful image IMG1, at limit values of the pixel inclusion ratio. The most important thing to see here is the closeness of the two limits, showing that ratio P (inside the studied interval) does not really affect the accuracy of the color reduction. The latter fact is confirmed in Fig. 9, which indicates the variation of ACD with respect to pixel inclusion ratio P for the same two images and various cluster count c. The difference between the input and output images decreases as the number of clusters grows. If we create 256-color palette for a typical image, the average color distance between input and output will be below 5 intensity units on the 0-255 scale.

Fig. 7. Some examples with input images in the top row, result images with reduced colors in bottom row: (a) colorful image IMG2 reduced to 48 colors, (b) colorful image IMG3 reduced to 96 colors, (c) typical image IMG5 reduced to 48 colors, (d) colorful image IMG4 reduced to 32 colors, (e) typical image IMG6 reduced to 48 colors, (f) image IMG8 reduced to 64 colors.

Another quantitative test of accuracy could be possible by comparing the cluster prototypes obtained by the proposed quick method with the ones given by the conventional slow HCM clustering algorithm. The numerical evaluation criterion is given as:

$$E = \sum_{i=1}^{c} ||\mathbf{v}_i - \mathbf{v}_i^{(\mathrm{conv})}|| \ , \tag{5}$$

where $\mathbf{v}_i^{(\mathrm{conv})}$ is the cluster prototype given by the conventional HCM clustering which best corresponds to $\mathbf{v}_i$. The smaller the value of E, the better approximation is given by the proposed method. Providing such an evaluation stands beyond the scope of this paper.

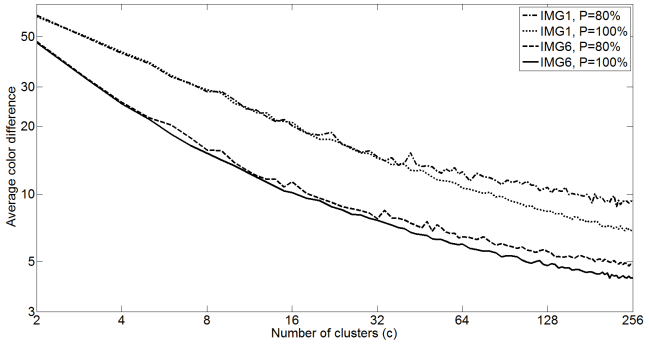

Fig. 8. Average color difference vs. cluster number c

V. DISCUSSION

The first and last stage of the processing, namely the histogram extraction and generation of output image, respectively, have the theoretical complexity of $\mathcal{O}(n)$, where n represents the number of pixels in the image. The second stage, which performs the clustering of aggregated colors, is $\mathcal{O}(cQ)$-complex, where $Q < nrCol$ is the number of those aggregated colors that were selected for the clustering process. Earlier we have seen that the total runtime does not grow proportionally with the number of input pixels, and neither with the number of clusters. There exist settings,

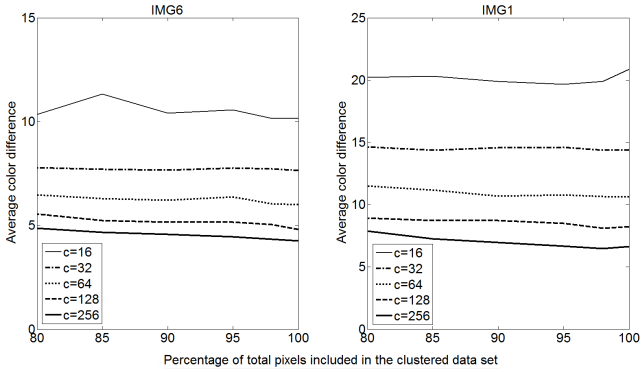

Fig. 9. Average color difference vs. pixel inclusion ratio P

when one of these parameters is virtually proportional with total runtime, but never do both. When the input image has several millions of pixels, but the number of aggregated colors selected for clustering is low, the 1st and 3rd stage of the proposed algorithm require the majority of the total runtime. Alternately, when the input image is small but colorful, and especially when the chosen number of clusters is also high, the 2nd stage of the algorithm will be dominant in terms of computational load. Table III gives us some hints in this sense, showing the time necessary for the 1st and 3rd, and the 2nd stage separately, in various scenarios, using HCM based clustering for the color reduction of a typical image. The 1st and 3rd stage roughly need constant time for the same image, they hardly depend on parameters like c and P. On the other hand, the execution time of the 2nd stage is strongly influenced by the same parameters.

During the numerical evaluation of the proposed methodology we concentrated on obtaining accurate results. HCM reaches convergence right after the first iteration in which no changes happen in the partition. During our tests this usually occurred after 30-70 iterations. However, we maximized the number of performed loops at 50, considering that the changes occurred thereafter would hardly affect the output image. Earlier solutions (e.g. [5]) suggested stopping after 20 iterations in order to reduce the execution time.

TABLE III

DURATION (IN $msec$) OF $\mathcal{O}(n)$-COMPLEX 1ST AND 3RD STAGE AND $\mathcal{O}(cQ)$-COMPLEX 2ND STAGE OF THE HCM BASED ALGORITHM APPLIED TO A TYPICAL IMAGE, IN CASE OF CLUSTER NUMBER $c \in \{32, 128\}$ AND VARIOUS PIXEL INCLUSION RATIOS P

IMG6 at resolution	800×600		1200×900		1600×1200		2048×1536		2892×2169		3648×2736	
Cluster count c	32	128	32	128	32	128	32	128	32	128	32	128
1st and 3rd stage	11		27		50		86		171		280	
2nd stage at $P = 100\%$	21	66	30	93	34	112	38	120	40	135	42	154
2nd stage at $P = 98\%$	8	18	9	20	10	20	11	25	14	27	15	31
2nd stage at $P = 95\%$	7	15	9	14	9	15	10	19	11	20	12	22

The efficiency tests performed in this study revealed the superiority (speed-up ratio in range 2-3) of the proposed method compared with the recent solution given in [5].

Some advantages of the proposed methodology are listed below:

1) It can be applied to reduce the number of colors in an image to virtually any integer number $c \geq 2$. This means higher flexibility compared to simply cutting least significant bits in every channel of the image.
2) There is space for further improvement of execution time via parallel computation, which can be easily employed in the $\mathcal{O}(n)$-complex stages of the algorithm, when performed on a multi-core PC.
3) The proposed framework is suitable to employ any kind of c-means clustering algorithm. For example, some generalized suppressed FCM algorithms [7], [18] tend to favor clusters of reduced size and are close to HCM to run sufficiently fast.

Limitations of the proposed method include:

1) It is well known that the initialization of the cluster prototypes can strongly influence the results of c-means clustering [1]. We need to employ a static procedure, which quickly provides 3D grid points that cover the whole color space as much as possible. Context sensitive intelligent procedures are prohibitively slow.
2) A few percent of the pixels are ignored by the color clustering process. This may slightly shift optimal cluster prototypes, which is a price to pay to get rid of a significant part of computational burden.
3) A fully automated optimal color reduction procedure would also need a mean to automatically find the right number of cluster for each image.

VI. CONCLUSIONS

In this paper we have introduced a fast color reduction algorithm based on c-means clustering. The proposed methods first performs a static color quantization followed by a histogram based color selection scheme and employs HCM to extract optimal reduced colors. The algorithm uses as key parameter the ratio of pixels included into the clustering process. Dropping properly chosen $2-5\%$ of the input data can significantly reduce execution time without losing important details. Numerical evaluation proved the superiority of the proposed methodology over recent solutions in the field. The proposed algorithm allows for parallel implementation of key parts, which can further reduce execution time without losing any part of the accuracy. Further works will aim at getting benefit from parallel execution.

REFERENCES

[1] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," *Proc. 18th Ann. ACM-SIAM Symp. Discr. Alg.*, 2007, pp. 1027–1035.

[2] A. Atsalakis, and N. Papamarkos, "Color reduction and estimation of the number of dominant colors by using a self-growing and self-organized neural gas," *Engineering Applications of Artificial Intelligence*, vol. 19, pp. 769–786, 2006.

[3] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*, New York: Plenum, 1981.

[4] J. P. Braquelaire and L. Brun, "Comparison and optimization of methods of color image quantization," *IEEE Transactions on Image Processing* vol. 6, pp. 1048–1052, 1997.

[5] M. E. Celebi, "Improving the performance of k-means in color quantization," *Image and Vision Computing*, vol. 29, pp. 260–271, 2011.

[6] G. Dong and M. Xie, "Color clustering and learning for image segmentation based on neural networks," *IEEE Transactions on Neural Networks* vol. 16, pp. 925–936, 2005.

[7] J. L. Fan, W. Z. Zhen and W. X. Xie, "Suppressed fuzzy c-means clustering algorithm," *Pattern Recognition Letters*, vol. 24, pp. 1607–1612, 2003.

[8] A. T. Ghanbarian, E. Kabir and N. M. Charkari, "Color reduction based on ant colony," *Pattern Recognition Letters*, vol. 28, pp. 1383–1390, 2007.

[9] I. S. Hsieh, K. C. Fan, "An adaptive clustering algorithm for color quantization," *Pattern Recognition Letters*, vol. 21, pp. 337–346, 2000.

[10] K. Kanjanawanishkul and B. Uyyanonvara, "Novel fast color reduction algorithm for time-constrained applications," *Journal of Visual Communication and Image Representation* vol. 16, pp. 311–332, 2005.

[11] J. F. Kolen and T. Hutcheson, "Reducing the time complexity of the fuzzy c-means algorithm," *IEEE Transactions on Fuzzy Systems* vol. 10, pp. 263–267, 2002.

[12] A. Mojsilovic and E. Soljanin, "Color quantization and processing by Fibonacci lattices ," *IEEE Transactions on Image Processing* vol. 10, pp. 1712–1725, 2001.

[13] N. Nikolaou, and N. Papamarkos, "Color reduction for complex document images," *International Journal of Imaging Systems and Technologies*, vol. 19, pp. 14–26, 2009.

[14] D. Özdemir, and L. Akarun, "A fuzzy algorithm for color quantization of images," *Pattern Recognition*, vol. 35, pp. 1785–1791, 2002.

[15] J. Rasti, A. Monadjemi and A. Vafaei, "Color reduction using a multi-stage Kohonen self-organizing map with redundant features," *Expert Systems with Applications*, vol. 38, pp. 13188–13197, 2011.

[16] H. Steinhaus, "Sur la division des corp materiels en parties", *Bulletin de l'Academie Polonaise des Science*, C1 III., vol. IV, pp. 801–804, 1956.

[17] L. Szilágyi, Z. Benyó, S. M. Szilágyi and H. S. Adam, "MR brain image segmentation using an enhanced fuzzy c-means algorithm," *Annual International Conference of IEEE EMBS,* Cancún, pp. 724–726, 2003.

[18] L. Szilágyi, S. M. Szilágyi and Cs. Kiss, "A generalized approach to the suppressed fuzzy c-means algorithm," *Lecture Notes in Computer Science*, vol. 6408, pp. 140–151, 2010.