

# An Efficient Sparse Data Filtering Method for Image Histogram Comparison

Marius Tico, Pauli Kuosmanen  
Digital Media Institute,  
Signal Processing Laboratory,  
Tampere University of Technology,  
P.O.BOX 553, FIN-33101, Tampere, FINLAND

## Abstract

The traditional method for creating a color histogram is to equally subdivide a certain color space into a given number of bins and then count the number of pixels each bin contains. Using such a histogram, the performance of almost all methods of histogram comparison are highly sensitive to the localization of the bin boundaries. In order to alleviate this sensitivity a new method of histogram creation is proposed in this paper. The histogram at a certain resolution (number of bins) is obtained in two steps. First a low pass filter is applied onto the highest resolution histogram and then the filtered histogram is quantized into the required number of bins. Based on the sparse nature of the color histogram an efficient algorithm which concomitantly performs both steps is proposed. The algorithm does not require the prior creation of the highest resolution histogram. It creates the required histogram directly in a single scan of the original image by counting the contributions of each image pixel to different histogram bins. The experimental results reveal a significant improvement of image retrieval performances when the histograms are created using the proposed method over the case they are created using the traditional approach.

## 1 Introduction

The most popular technique for image retrieval in a heterogeneous collection of images is the comparison of images based on their histograms. The histogram describes the gray-level or color distribution for a given image. It is a global feature which can be used to perform a fast but no so reliable indexing process. The histogram feature can be used as a preliminary step for database indexing in order to reduce the number of candidate images for the next step which could use other features (e.g. shape, texture, orientation) to compare the database images with a given query image. The major advantage offered by the histogram feature con-

sists in its small sensitivity to scale, rotation and translation [1]. An appropriate color space, a color quantization scheme, a histogram representation, and a similarity metric are the main elements required for the design of a histogram based retrieval system [2].

The RGB color space is inappropriate for image retrieval due to the fact that it is not related with the way humans perceive colors. Other color spaces like *opponent color space*, HSV and YIQ are generally used for retrieval proposes [2][3]. The Lu<sup>\*v\*</sup> space is also used because it yields a perceptually uniform spacing of colors [4].

The total number of different colors that can be represented using a resolution of 8 bits for each primary component is about 16.8 million. Such a huge amount of data is difficult to store and process. Nevertheless, lower resolution representations, obtained by quantizing each primary component on a smaller number of bits are used for histogram representation. A quantized histogram is usually represented as an  $M$  dimensional vector, where  $M$  is the total number of bins (quantization intervals). For example a resolution of 4 bits for hue, 3 bits for saturation and 3 bits for value meaning a total number of 1024 bins, seems to be enough for obtaining good retrieval performance, as is shown in [2].

Many similarity metrics between image histograms have been proposed. Based on the hypothesis that similar images will have similar color distributions a histogram intersection metric to measure the similarity of two histograms is proposed in [1]. For two given histograms represented by two  $M$  dimensional column vectors  $A$  and  $B$  the histogram intersection operator ( $IO$ ) is given by:

$$IO(A, B) = \sum_{i=0}^{M-1} \min(A(i), B(i)), \quad (1)$$

where  $A(i)$  and  $B(i)$  represents the number of pixels of  $A$  and  $B$  images which occur in the bin  $i$ . Usually the number of pixels in each bin is divided by the total number of pixels in the image to obtain an estimation

Table 1: The results of indexing A and B images based on the query image Q using the intersection operator at different histogram resolutions.

n	$Q_n$	$A_n$	$B_n$	$IO(Q_n, A_n)$	$IO(Q_n, B_n)$	order
0	(0000000100000000)	(0000000010000000)	(0000100000000000)	0	0	-
1	(00010000)	(00001000)	(00100000)	0	0	-
2	(0100)	(0010)	(0100)	0	1	$B, A$
3	(10)	(01)	(10)	0	1	$B, A$
4	(1)	(1)	(1)	1	1	-

Table 2: The results of indexing A and B images based on the query image Q using the intersection operator at different histogram resolutions if the high resolution histogram is filtered before quantization.

n	$Q_n$	$A_n$	$B_n$	$IO(Q_n, A_n)$	$IO(Q_n, B_n)$	order
0	$(00000 \frac{1}{5} \frac{1}{5} \frac{1}{5} \frac{1}{5} \frac{1}{5} 000000)$	$(000000 \frac{1}{5} \frac{1}{5} \frac{1}{5} \frac{1}{5} \frac{1}{5} 00000)$	$(00 \frac{1}{5} \frac{1}{5} \frac{1}{5} \frac{1}{5} \frac{1}{5} 000000000)$	$\frac{4}{5}$	$\frac{2}{5}$	$A, B$
1	$(00 \frac{1}{5} \frac{2}{5} \frac{2}{5} 000)$	$(000 \frac{2}{5} \frac{2}{5} \frac{1}{5} 00)$	$(0 \frac{2}{5} \frac{2}{5} \frac{1}{5} 0000)$	$\frac{4}{5}$	$\frac{2}{5}$	$A, B$
2	$(0 \frac{3}{5} \frac{2}{5} 0)$	$(0 \frac{2}{5} \frac{3}{5} 0)$	$(\frac{2}{5} \frac{3}{5} 00)$	$\frac{4}{5}$	$\frac{2}{5}$	$A, B$
3	$(\frac{3}{5} \frac{2}{5})$	$(\frac{2}{5} \frac{3}{5})$	(10)	$\frac{4}{5}$	$\frac{2}{5}$	$A, B$
4	(1)	(1)	(1)	1	1	-

of the color probability mass function. The larger the intersection value the more similar the histograms.

Another similarity metric which takes into account the perceptual similarity between bins is the cross distance function (CDF) proposed in [5]:

$$\begin{aligned}
 CDF(A, B) &= (A - B)^t \mathbf{C} (A - B) = \\
 &= \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} c_{ij} (A(i) - B(i))(A(j) - B(j)), \quad (2)
 \end{aligned}$$

where the matrix  $\mathbf{C} = \{c_{ij}\}$  stores the similarity coefficients between colors corresponding to bins  $i$  and  $j$ . The high computational complexity of the cross distance function  $\mathcal{O}(M^2)$  makes it a less efficient technique than the histogram intersection operator ( $IO$ ), especially when dealing with large image databases.

On the other hand, the  $IO$  similarity metric is highly dependent of the location of the quantization boundaries and hence it could fail in most of the cases due to the sparse nature of the image histograms. A simple example is presented next in order to clarify this problem.

Let  $Q$  denotes the histogram of a certain query image. Also, let  $A$  and  $B$  denote the histograms of two images from a certain collection of images. It is assumed, for simplicity, that the images are represented on 16 gray levels and each one of them represents a constant gray level. For example, all the pixels in the image  $Q$  have the gray level 7, all the pixels in the image  $A$  have the gray level 8, and all the pixels in the image  $B$  have the gray level 4. Obviously, from the histogram comparison point of view, the image  $A$  is “closer” to the image  $Q$  than it is the image  $B$ . Consequently, the image  $A$  has

to occur before the image  $B$  in the indexed database. Formally, this is expressed by writing that the order is equal with  $A, B$  rather than  $B, A$ . Nevertheless, the results obtained by using the intersection operator at different histogram resolutions are not reflecting the true situation as shown in Table 1. There the symbol  $X_n$  means the histogram of the image  $X$  represented at the resolution  $N - n$  ( $2^{N-n}$  bins), where  $N$  is the number of bits required to represent the highest resolution histogram. In our example  $N = 4$  and  $n = 0, 1, 2, 3, 4$ . The last column of the Table 1 suggests the order of the images  $A$  and  $B$  in the indexed database. A “-” in a certain row of this column signifies that the intersection operator fails in determining a certain indexing order of the images  $A$  and  $B$ . It gives the same result when comparing  $Q$  with  $A$  as when comparing  $Q$  with  $B$ .

Inspecting the Table 1, one can easily observe that the intersection operator either fails or determines a wrong indexing order of the images  $A$  and  $B$ . This result is primarily due to the high sensitivity of the intersection operator to the localization of the bin boundaries.

A possibility to overcome this problem consists in applying a linear low pass filter on to the highest resolution histogram before its quantization at different resolutions. In this way each gray level is spread in more than one bins, being forced to contribute not only to its bin but also to the neighborhood bins. Continuing the above example, Table 2 shows the results obtained when the highest resolution histograms are filtered with a symmetric 5 taps filter  $f(0) = f(\pm 1) = f(\pm 2) = 1/5$  before quantizing them at different resolutions. The correct indexing order is determined now for almost any

resolution of histogram representation.

Although, the above example refers only the case of gray level images, similar situations occur also when dealing with color images. The main difference is that a color histogram estimates the 3D color probability mass function of the image, and therefore the bins becomes parallelepipeds when each color component is subdivided in a certain number of intervals.

A direct application of the method of histogram creation suggested above is, however, almost impossible in the case of color histograms. This is because the highest resolution color histogram achieves an enormous size, e.g.  $2^{24}$  bins. Therefore, the difficulty consist not only in filtering such a huge signal but even in storing it in the memory.

This paper presents a method of filtered histogram creation which, exploiting the sparseness of the color histogram, succeeds to avoid the above mentioned difficulties. The method does not required the prior creation of the highest resolution histogram but instead the filtered histogram at a given resolution is computed by adding the weighted contributions of each image pixel to different histogram bins. The filtered histogram is thereby computed in a single scan of the original image and the algorithm achieves a low complexity of  $\mathcal{O}(qP)$  where  $P$  is the number of pixels in the image and  $q$  is the number of bins affected by the contributions of a single pixel.

The remaining of the paper is organized as follows. An efficient method of 1D filtered histogram creation is presented in the Section 2. The extension of the method for the case of color histograms is then presented in the Section 3. The experimental results presented in Section 4 compare the proposed histogram creation method with the traditional method of histogram creation, based on their retrieval performances in a heterogeneous collection of images. In addition Section 4 includes also the concluding remarks of the paper.

## 2 The Creation of a 1D Filtered Histogram

Let  $x(i), i = 0, \dots, 2^N - 1$ , denotes the highest resolution histogram of a certain gray level image. The filtered histogram  $y(j), j = 0, \dots, 2^N - 1$ , is obtained from  $x$  by using a filter  $f(k), k = -K, \dots, K$  as follows:

$$y(j) = (f \star x)(j) = \sum_{k=-K}^K f(k)x(j-k), \quad (3)$$

where  $y$  is considered to be equal to zero outside  $j \in \{0, \dots, 2^N - 1\}$ .

Next, an intermediate signal denoted by  $z_n(j), j =$

$0, \dots, 2^N - 1$ , is defined as below:

$$z_n(j) = \sum_{i=0}^{2^n-1} y(j+i). \quad (4)$$

The signal  $z_n$  can be written also in the form:

$$z_n(j) = \sum_{i=-2^n+1}^0 g_n(i)y(j-i) = (g_n \star y)(j), \quad (5)$$

where  $g_n(j) = 1$  if  $j \in \{-2^n + 1, \dots, 0\}$  and zero otherwise, and  $z_n$  is restricted to  $2^N$  elements by considering  $z_n(j) = 0$  if  $j \notin \{0, \dots, 2^N - 1\}$ .

The relation between  $z_n$  and  $x$  is:

$$\begin{aligned} z_n(j) &= (g_n \star f \star x)(j) = ((g_n \star f) \star x)(j) = \\ &= (h_n \star x)(j), \end{aligned} \quad (6)$$

where

$$h_n(j) = (g_n \star f)(j). \quad (7)$$

The filter  $h_n(j)$  has  $2^n + 2K$  nonzero taps and it is zero for  $j \notin \{-2^n - K + 1, \dots, K\}$ . This filter can be pre-computed and stored before processing the histogram.

Let us denote by  $x_n(i), i = 0, \dots, 2^{N-n} - 1$ , the filtered image histogram at the resolution  $N - n$  obtained by representing the highest resolution filtered histogram  $y$  onto  $2^{N-n}$  bins. One can note that the signal  $x_n$  can be obtained from  $z_n$  by sub-sampling the last one with  $2^n$

$$x_n(i) = z_n(i \cdot 2^n). \quad (8)$$

The above method for the computation of the quantized and filtered histogram  $x_n(i)$  requires the prior creation of the highest resolution image histogram. This is not a problem in the case of gray-level images where the highest resolution histogram usually has a dimension of 256 bins. Nevertheless, as explained in the introduction, the creation of the highest resolution histogram constitutes an insurmountable problem in the case of color images especially because of its huge dimensionality (e.g.  $2^{24}$  bins when each color component is represented on 8 bits).

The approach to avoid the creation of the highest resolution histogram consists in the computation of the filtered histogram at a certain resolution by adding the contributions of each image pixel to different bins rather than applying the filter directly in the color space.

One can show from Eq. 6, that each  $x(j)$  contributes to  $2^n + 2K$  different  $z_n$  taps. This contribution is expressed as:

$$z_n(j+i) \leftarrow z_n(j+i) + h_n(i)x(j). \quad (9)$$

From Eq. 8 we see that each  $x(j)$  will contribute to only at most  $q = \lceil 1 + K \cdot 2^{-n+1} \rceil$  of  $x_n$  taps:

$$x_n(i) \leftarrow x_n(i) + h_n(i \cdot 2^n - j)x(j), \quad (10)$$

where  $\lceil a \rceil$  denotes the smallest integer greater than or equal to the real number  $a$ . For example, if  $n = 6$  and  $K = 10$  only one or at most two contributions of type Eq. 10 has to be computed for each  $x(j)$  and for  $n = 6$ ,  $K = 100$  the number of contributions becomes at most 5.

The computation efficiency can be further improved if the  $h_n$  taps are arranged in a certain order. Denote by  $\mathbf{H}_n$  a  $q \times 2^n$  matrix, constructed as follows:

$$\mathbf{H}_n = \begin{pmatrix} \tilde{h}_n(0) & \dots & \tilde{h}_n(2^n - 1) \\ \tilde{h}_n(2^n) & \dots & \tilde{h}_n(2^n + 2^n - 1) \\ \vdots & \dots & \vdots \\ \tilde{h}_n((q-1) \cdot 2^n) & \dots & \tilde{h}_n((q-1) \cdot 2^n + 2^n - 1) \end{pmatrix}, \quad (11)$$

where  $q = \lceil 1 + K \cdot 2^{-n+1} \rceil$  and  $\tilde{h}_n$  is a  $2K + 2^n$  taps causal filter obtained by translating the filter  $h_n$ :

$$\tilde{h}_n(i) = h_n(-2^n - K + 1 + i), \quad (12)$$

where  $0 \leq i \leq 2K + 2^n - 1$ . The contributions of any input  $x(j)$  with gray level  $j$  to different  $x_n$  taps are weighted by the values of only a single column of matrix  $\mathbf{H}_n$ . On the other hand, all contributions received in a certain bin  $i$ , which are accumulate to  $x_n(i)$  are weighted by the values of a single row of the matrix. Let  $c$  and  $r$  denote the column and row of  $\mathbf{H}_n$  corresponding to a gray level  $j$  and a bin  $i$ , respectively. From Eq. 10 we obtain:

$$\mathbf{H}_n(r, c) = h_n(i \cdot 2^n - j). \quad (13)$$

From Eq. 11 and Eq. 12 we have:

$$\begin{aligned} \mathbf{H}_n(r, c) &= \tilde{h}_n((r-1) \cdot 2^n + (c-1)) = \\ &= h_n(-2^n - K + 1 + (r-1) \cdot 2^n + (c-1)). \end{aligned} \quad (14)$$

The following expression between  $r, c, i$  and  $j$  results from Eq. 13 and Eq. 14:

$$c - 1 = (i + 1 - (r - 1)) \cdot 2^n + K - j - 1. \quad (15)$$

Furthermore using the notation:

$$K - j - 1 = \left\lfloor \frac{K - j - 1}{2^n} \right\rfloor \cdot 2^n + (K - j - 1) \bmod 2^n, \quad (16)$$

we obtain:

$$\begin{aligned} c - 1 &= \left( i + 1 - (r - 1) + \left\lfloor \frac{K - j - 1}{2^n} \right\rfloor \right) \cdot 2^n + \\ &+ (K - j - 1) \bmod 2^n, \end{aligned} \quad (17)$$

where  $\lfloor a \rfloor$  denotes the greatest integer equal or smaller than the real number  $a$ . Considering that  $0 \leq c - 1 \leq 2^n - 1$ , we obtain from Eq. 17:

$$\begin{aligned} c &= (K - j) \bmod 2^n, \\ r &= i + \left\lfloor \frac{K - j - 1}{2^n} \right\rfloor + 2. \end{aligned} \quad (18)$$

The last two expressions simplify the access to different taps of the filter  $h_n$  and can be used to implement an efficient algorithm for the creation of the filtered histogram at a certain resolution directly from the input image. The algorithm is shown in Fig.1 and its complexity is  $\mathcal{O}(qP)$ , where  $P$  is the total number of pixels in the image and  $q$  is the maximum number of contributions computed for each pixel.

#### PROCEDURE 1D-Histogram-Creation

**INPUT:**

$I$  -the image  
 $n, K$   
 $\mathbf{H}_n$  -precomputed matrix

**OUTPUT:**

$x_n(i)$ ,  $0 \leq i \leq 2^{N-n} - 1$  - the filtered histogram

$q = \lceil 1 + K \cdot 2^{-n+1} \rceil$

$x_n(i) = 0$  for  $0 \leq i \leq 2^{N-n} - 1$

**FOR** each image pixel  $p \in I$  with the gray level  $j$  **DO**

$c = (K - j) \bmod 2^n$

**FOR**  $r = 1 \dots q$  **DO**

$i = r - \left\lfloor \frac{K - j - 1}{2^n} \right\rfloor - 2$

$x_n(i) \leftarrow x_n(i) + \mathbf{H}_n(r, c)$

**END FOR**

**END FOR**

**END PROCEDURE**

Figure 1: The algorithm for the computation of the filtered histogram ( $x_n$ ) directly from the input image. The prior creation of the highest resolution histogram ( $x = x_0$ ) is not required.

## 2.1 Computing a Lower Resolution Filtered Histogram from a Higher Resolution One

After the computation of a filtered histogram  $x_n$  at a given resolution, any other filtered histogram at a lower resolution can be obtained immediately from  $x_n$  without applying the filter again. From Eq. 8 and Eq. 4 we have:

$$x_{n+1}(i) = z_{n+1}(i \cdot 2^{n+1}) = \sum_{j=0}^{2^{n+1}-1} y(i \cdot 2^{n+1} + j). \quad (19)$$

The right sum in Eq. 19 can be also written as:

$$\begin{aligned} \sum_{j=0}^{2^{n+1}-1} y(i \cdot 2^{n+1} + j) &= \sum_{j=0}^{2^n-1} y(2i \cdot 2^n + j) + \\ &+ \sum_{j=0}^{2^n-1} y((2i+1) \cdot 2^n + j). \end{aligned} \quad (20)$$

Using Eq. 4 again in Eq. 20 and combining the result with Eq. 19, a simple relation between a filtered histogram at two different resolutions is derived:

$$x_{n+1}(i) = x_n(2i) + x_n(2i + 1). \quad (21)$$

Therefore, once a filtered histogram at a certain resolution is computed, its representation at any other coarser resolution can be obtained without applying the filtering procedure again.

### 3 The Creation of a 3D Filtered Histogram

The filtered histogram creation algorithm presented in the previous section is extended here to the case of color histograms.

Consider a three-dimensional separable filter  $f$ :

$$f(k_1, k_2, k_3) = f_1(k_1)f_2(k_2)f_3(k_3), \quad (22)$$

where  $-K_p \leq k_p \leq K_p$ , for  $p \in \{1, 2, 3\}$ . The three-dimensional highest resolution histogram has a resolution of  $2^{N_1} \times 2^{N_2} \times 2^{N_3}$  and it is denoted by  $x(i_1, i_2, i_3)$ , where  $0 \leq i_1 < 2^{N_1}, 0 \leq i_2 < 2^{N_2}, 0 \leq i_3 < 2^{N_3}$ . Filtering this histogram with the filter  $f$ , a blurred version of the three dimensional histogram is obtained:

$$y(j_1, j_2, j_3) = (f \star_3 x)(j_1, j_2, j_3), \quad (23)$$

where  $\star_3$  denotes the three dimensional convolution. The signal  $y(j_1, j_2, j_3)$  is set to zero for  $j_p < 0$  or  $j_p \geq 2^{N_p}$ , where  $p \in \{1, 2, 3\}$ . A three-dimensional, intermediate signal is obtained using a relation similar with Eq. 4:

$$\begin{aligned} & z_{n_1, n_2, n_3}(j_1, j_2, j_3) = \\ & = \sum_{i_1=0}^{2^{n_1}-1} \sum_{i_2=0}^{2^{n_2}-1} \sum_{i_3=0}^{2^{n_3}-1} y(j_1 + i_1, j_2 + i_2, j_3 + i_3), \end{aligned} \quad (24)$$

which is further equivalent as in Eq. 5 with:

$$\begin{aligned} & z_{n_1, n_2, n_3}(j_1, j_2, j_3) = \\ & \sum_{i_1, i_2, i_3} g_{n_1, n_2, n_3}(i_1, i_2, i_3) y(j_1 - i_1, j_2 - i_2, j_3 - i_3), \end{aligned} \quad (25)$$

where  $g_{n_1, n_2, n_3}$  is a three dimensional separable filter

$$g_{n_1, n_2, n_3}(i_1, i_2, i_3) = g_{n_1}(i_1)g_{n_2}(i_2)g_{n_3}(i_3), \quad (26)$$

with

$$g_{n_p}(i_p) = \begin{cases} 1 & -2^{n_p} + 1 \leq i_p \leq 0, \\ 0 & \text{otherwise,} \end{cases} \quad (27)$$

for  $p \in \{1, 2, 3\}$ . Using Eq. 23 and Eq. 25, the intermediate signal can be written as:

$$z_{n_1, n_2, n_3} = ((g_{n_1, n_2, n_3} \star_3 f) \star_3 x) = (h_{n_1, n_2, n_3} \star_3 x), \quad (28)$$

where  $h_{n_1, n_2, n_3}$  is a three dimensional separable filter:

$$h_{n_1, n_2, n_3}(j_1, j_2, j_3) = h_{n_1}^{(1)}(j_1)h_{n_2}^{(2)}(j_2)h_{n_3}^{(3)}(j_3), \quad (29)$$

where:

$$h_{n_p}^{(p)}(j_p) = (g_{n_p} \star f_p)(j_p), \quad p \in \{1, 2, 3\}. \quad (30)$$

Each filter  $h_{n_p}^{(p)}(j_p)$  is a one dimensional filter with  $2^{n_p} + 2K_p$  nonzero taps, being zero for  $j_p \notin \{-2^{n_p} - K_p + 1, \dots, K_p\}$ , where  $p \in \{1, 2, 3\}$ . These three filters can be precomputed and stored before filtering the histogram.

The three dimensional histogram  $x_{n_1, n_2, n_3}(i_1, i_2, i_3)$ ,  $0 \leq i_p < 2^{N_p - n_p}$ ,  $p \in \{1, 2, 3\}$  with a resolution of  $2^{N_1 - n_1} \times 2^{N_2 - n_2} \times 2^{N_3 - n_3}$  is then obtained by sub-sampling the three dimensional signal  $z_{n_1, n_2, n_3}$  as follows:

$$\begin{aligned} & x_{n_1, n_2, n_3}(i_1, i_2, i_3) = \\ & = z_{n_1, n_2, n_3}(i_1 \cdot 2^{n_1}, i_2 \cdot 2^{n_2}, i_3 \cdot 2^{n_3}). \end{aligned} \quad (31)$$

From Eq. 28 we see that each tap  $x(j_1, j_2, j_3)$  contributes to  $(2^{n_1} + 2K_1) \times (2^{n_2} + 2K_2) \times (2^{n_3} + 2K_3)$  taps of  $z_{n_1, n_2, n_3}$ . These contributions can be expressed in a similar form with Eq. 9 as:

$$\begin{aligned} & z_{n_1, n_2, n_3}(j_1 + i_1, j_2 + i_2, j_3 + i_3) \leftarrow \\ & \leftarrow z_{n_1, n_2, n_3}(j_1 + i_1, j_2 + i_2, j_3 + i_3) + \\ & + h_{n_1}^{(1)}(i_1)h_{n_2}^{(2)}(i_2)h_{n_3}^{(3)}(i_3)x(j_1, j_2, j_3), \end{aligned} \quad (32)$$

where the separability property of the three dimensional filter  $h_{n_1, n_2, n_3}$  has been used.

Because the filtered histogram  $x_{n_1, n_2, n_3}$  is a sub-sampled version of the signal  $z_{n_1, n_2, n_3}$ , from Eq. 31 it follows that each tap  $x(j_1, j_2, j_3)$  contributes to  $q_1 \times q_2 \times q_3$  taps of  $x_{n_1, n_2, n_3}$ . These contributions are expressed in a relation similar with Eq. 10:

$$\begin{aligned} & x_{n_1, n_2, n_3}(i_1, i_2, i_3) \leftarrow x_{n_1, n_2, n_3}(i_1, i_2, i_3) + \\ & + x(j_1, j_2, j_3) \prod_{p=1}^3 h_{n_p}^{(p)}(i_p \cdot 2^{n_p} - j_p) \end{aligned} \quad (33)$$

and the values  $q_1, q_2, q_3$  are given by:

$$q_p = \lceil 1 + K_p \cdot 2^{-n_p + 1} \rceil, \quad p \in \{1, 2, 3\}. \quad (34)$$

The algorithm shown in Fig.2, uses the separability property of the filter  $h_{n_1, n_2, n_3}$  to speed up the access time to different taps of this filter. Three matrixes  $\mathbf{H}_{n_1}^{(1)}$ ,  $\mathbf{H}_{n_2}^{(2)}$  and  $\mathbf{H}_{n_3}^{(3)}$  corresponding to the three one-dimensional filters  $h_{n_1}^{(1)}, h_{n_2}^{(2)}$  and  $h_{n_3}^{(3)}$ , respectively, are constructed as in Eq. 11.

The algorithm shown in Fig.2 achieves a complexity of  $\mathcal{O}(q_1 q_2 q_3 P)$ , where  $P$  is the total number of pixels in the image and  $q_1 q_2 q_3$  is the maximum number of contributions computed for each pixel.

## PROCEDURE 3D-Histogram-Creation

**INPUT:**

$I$  -the image

$n_1, n_2, n_3, K_1, K_2, K_3$

$\mathbf{H}_{n_1}^{(1)}, \mathbf{H}_{n_2}^{(2)}, \mathbf{H}_{n_3}^{(3)}$  -precomputed matrixes

**OUTPUT:** the 3D filtered histogram

$x_{n_1, n_2, n_3}(i_1, i_2, i_3), 0 \leq i_p \leq 2^{N_p - n_p} - 1, p \in \{1, 2, 3\}$

**Initialization**

$q_p = \lceil 1 + K_p \cdot 2^{-n_p + 1} \rceil$  for  $p = 1, 2, 3$

$x_{n_1, n_2, n_3}(i_1, i_2, i_3) = 0$  for  $0 \leq i_p \leq 2^{N_p - n_p} - 1$ , for each  $p \in \{1, 2, 3\}$

**FOR** each image pixel  $(j_1, j_2, j_3) \in I$  **DO**

$c_1 = (K_1 - j_1) \bmod 2^{n_1}$

$c_2 = (K_2 - j_2) \bmod 2^{n_2}$

$c_3 = (K_3 - j_3) \bmod 2^{n_3}$

**FOR**  $r_1 = 1 \dots q_1$  **DO**

$i_1 = r_1 - \left\lfloor \frac{K_1 - j_1 - 1}{2^{n_1}} \right\rfloor - 2$

**FOR**  $r_2 = 1 \dots q_2$  **DO**

$i_2 = r_2 - \left\lfloor \frac{K_2 - j_2 - 1}{2^{n_2}} \right\rfloor - 2$

**FOR**  $r_3 = 1 \dots q_3$  **DO**

$i_3 = r_3 - \left\lfloor \frac{K_3 - j_3 - 1}{2^{n_3}} \right\rfloor - 2$

$x_{n_1, n_2, n_3}(i_1, i_2, i_3) \leftarrow x_{n_1, n_2, n_3}(i_1, i_2, i_3) + \mathbf{H}_{n_1}^{(1)}(r_1, c_1) \mathbf{H}_{n_2}^{(2)}(r_2, c_2) \mathbf{H}_{n_3}^{(3)}(r_3, c_3)$

**END FOR**

**END FOR**

**END FOR**

**END FOR**

**END PROCEDURE**

Figure 2: The algorithm for the computation of the filtered 3D histogram.

## 4 Experimental Results and Conclusions

The proposed method of histogram creation has been compared with the traditional histogram creation approach. Their performances of retrieving similar images in a heterogeneous collection of images have been used as the comparison criterion. Based exclusively on color information different retrieval experiments have been done. In all cases the histograms created by either one of the methods were compared using as the similarity metric the intersection operator (*IO*) proposed by Swain and Ballard [1].

A heterogeneous collection of 2117 color images selected from a digital newspaper photo archive, has been used for experimental purposes. The images in the collection are quite representative for the general requirements of a newspaper. Therefore they represent various subjects, e.g. sports, politics, arts, fashion, travel, con-

certs, movies.

A number of eight images shown in Fig.3 have been artificially created by cutting and rescaling a given image from the database. It is assumed that these images are quite similar in color one to each other. Consequently, when any of these eight images is presented as a query we expect to retrieve the other seven images before any other image of the database. The eight images are labeled with  $I_1, \dots, I_8$  as is explained also in the caption of the Fig.3.

A query session indexes the images in the database according with their similarity with the query image. Therefore, the rank value of a certain image in the indexed database could be considered as an expression of the degree of dissimilarity between the given image and the query image. The lower the rank is the higher the similarity degree is. Based on the image diversity in our database we can consider that the rank value is a quite objective criteria to measure the similarity between the query image and a given image in the database.

We introduce here a retrieval performance measure called *Rank Offset (RO)* which quantitatively express how low the ranks of the relevant images are. This is, if the query image is one of the eight images then the ranks associated with the other seven relevant images should be as low as possible. Ideally they should be between 2 and 8 (the query image is the most similar image with itself and therefore it always has the rank 1). The Rank Offset is expressed as follows

$$RO = \sum_{i=1}^M (R_i - i), \quad (35)$$

where  $R_i$  is the rank of the  $i$ -th relevant image and  $M$  is the total number of relevant images. *RO* should be low. A zero value means that the relevant images are all in the top of retrieved images.

Besides the Rank Offset, we have used also four other retrieval performance measurements in order to compare the proposed histogram creation method with the traditional method of histogram creation. These performance measurements have been introduced in [2] and will be shortly explain in the following.

*Maximum Rank (MR)* is defined as the highest rank associated with one of the relevant images.

*Average Rank (AR)* is the average of the all rank values associated with the relevant images.

Both *AR* and *MR* should be low.

*Filtering Performance (FP)* is given in terms of the percentage of the relevant images which are retrieved. We consider that an image is retrieved if its rank is smaller or equal with 50. In other words, the image is in the top 50 images of the indexed database.

*Bull's Eye Performance (BEP)* is defined as the percentage of relevant images which are retrieved among



Figure 3: Eight images artificially created by cutting and rescaling a given image of the database. These are the images used for the experiments shown in the Tables 3, 4 and 5. The images are numbered from left to right and the label associated with the  $k$ -th image is denoted by  $I_k$ .

the top  $2M$  retrieved images, where  $M$  is the total number of relevant images.

The color space used in all our experiments was the *opponent color space* defined as:

$$\begin{aligned}
 wb &= R + G + B, \\
 rg &= R - G, \\
 by &= 2B - R - G.
 \end{aligned}
 \tag{36}$$

The results are shown in the Tables 3,4 and 5. The column  $Q$  specifies the query image used in each query session, and the columns called  $R_i$  express the rank in the indexed database associated with the  $i$ -th relevant image as a result of different query sessions.

Table 3 shows the retrieval performances achieved when the histograms have been created using the traditional approach. This is, the highest resolution histogram is not filtered before subdividing the color space in a certain number of bins. Tables 4 and 5 show the retrieval performances achieved when the histograms are created using the method proposed in this paper. The difference between the experiments shown in these two tables consists in the length of the filter used.

The comparison between the results shown in Tabel 3 with the results shown in the other two tables reveals a significant improvement in retrieval performance when the proposed method of histogram creation is used. For example, inspecting the Filtering Performance ( $FP$ ) one can note that only in three different query sessions all the eight relevant images have been retrieved among the top 50 images in the indexed database when the traditional method of histogram creation is used (see Table 3). On the other hand when the proposed method of histogram creation is used there are 6 and 7 different

query sessions when all the relevant images are retrieved among the top 50 images in the indexed database, as can be seen in Table 4 and Table 5 respectively.

It is of importance to note also that the comparison between Table 4 and Table 5 indicates improved performance via use of longer filter windows. The analysis of the effect of the size of the window and also of the filter tap values is left to another study.

## References

- [1] M.J.Swain, D.H.Ballard, "Color Indexing", *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11-32, 1991.
- [2] A.Vellaikal and C.C.J.Kuo, "Content based image retrieval using multiresolution histogram representation", *SPIE - Digital Image Storage and Archiving Systems*, vol. 2606, pp. 312-323, 1995.
- [3] H.J.Zhang, Y.Gong, C.Y.Low, S.W.Smoliar, "Image Retrieval based on Color Feature: An Evaluation Study", *SPIE - Digital Image Storage and Archiving Systems*, vol. 2606, pp. 212-220, 1995.
- [4] X.Wan, C.C.J.Kuo, "Pruned Octree Feature for Interactive Retrieval", *SPIE - Multimedia Storage and Archiving Systems II*, vol. 3229, pp. 182-193, 1997.
- [5] J.Hafner, H.S.Sawhney, W.Equitiz, M.Flickner, W. Niblack, "Efficient Color Histogram Indexing for Quadratic Form Distance Functions", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 17, no. 7, pp. 729-736, July 1995.

Table 3: The retrieval performances achieved when the histograms have been created using the traditional approach, by subdividing the color space in a certain number of bins without filtering the highest resolution histogram. Each opponent color space component is subdivided into 4 bins, and hence the color histogram of each image has a number of 64 bins.

$Q$	$R_1$	$R_2$	$R_3$	$R_4$	$R_5$	$R_6$	$R_7$	$R_8$	$MR$	$AR$	$RO$	$FP$ [%]	$BEP$ [%]
$I_1$	1	9	8	4	2	3	13	7	13	5.88	11	100.00	100.00
$I_2$	26	1	82	24	12	47	2	9	82	25.38	167	87.50	50.00
$I_3$	3	188	1	20	95	2	234	92	234	79.38	599	50.00	37.50
$I_4$	4	6	16	1	5	3	11	2	16	6.00	12	100.00	100.00
$I_5$	4	6	85	15	1	34	8	12	85	20.62	129	87.50	75.00
$I_6$	3	12	4	2	9	1	21	5	21	7.12	21	100.00	87.50
$I_7$	35	2	94	33	10	57	1	20	94	31.50	216	75.00	37.50
$I_8$	8	3	61	2	6	12	7	1	61	12.50	64	87.50	87.50
Average results									75.75	23.55	152.37	86.00	71.87

Table 4: The retrieval performances achieved when the histograms have been created using the proposed method. The highest resolution histograms have been filtered with the 3D separable filter:  $f_1 = f_2 = f_3 = f$ ,  $f(n) = 1/15$ ,  $n = -7 \dots 7$ . Each opponent color space component is subdivided in 4 bins, and hence the color histogram of each image has a number of 64 bins.

$Q$	$R_1$	$R_2$	$R_3$	$R_4$	$R_5$	$R_6$	$R_7$	$R_8$	$MR$	$AR$	$RO$	$FP$ [%]	$BEP$ [%]
$I_1$	1	3	5	4	2	8	7	9	9	4.88	3	100.00	100.00
$I_2$	5	1	49	11	3	33	2	10	49	14.25	78	100.00	75.00
$I_3$	3	40	1	4	18	2	65	14	65	18.38	111	87.50	62.50
$I_4$	4	5	6	1	7	3	14	2	14	5.25	6	100.00	100.00
$I_5$	4	3	33	10	1	35	2	16	35	13.00	68	100.00	75.00
$I_6$	5	9	4	2	13	1	39	3	39	9.50	40	100.00	87.50
$I_7$	19	2	53	25	3	46	1	30	53	22.38	143	87.50	37.50
$I_8$	5	4	24	2	11	3	18	1	24	8.50	32	100.00	75.00
Average results									36.00	12.02	60.12	96.87	76.56

Table 5: The retrieval performances achieved when the histograms have been created using the proposed method. The highest resolution histograms have been filtered with the 3D separable filter:  $f_1 = f_2 = f_3 = f$ ,  $f(n) = 1/31$ ,  $n = -15 \dots 15$ . Each opponent color space component is subdivided in 4 bins, and hence the color histogram of each image has a number of 64 bins.

$Q$	$R_1$	$R_2$	$R_3$	$R_4$	$R_5$	$R_6$	$R_7$	$R_8$	$MR$	$AR$	$RO$	$FP$ [%]	$BEP$ [%]
$I_1$	1	3	5	7	2	4	6	9	9	4.62	1	100.00	100.00
$I_2$	4	1	32	5	3	10	2	6	32	7.88	27	100.00	87.50
$I_3$	3	53	1	5	16	2	50	24	53	19.25	118	87.50	62.50
$I_4$	6	5	10	1	17	3	11	2	17	6.88	19	100.00	87.50
$I_5$	2	3	7	5	1	8	4	10	10	5.00	4	100.00	100.00
$I_6$	5	17	3	2	20	1	21	4	21	9.12	37	100.00	62.50
$I_7$	4	2	22	5	3	11	1	10	22	7.25	22	100.00	87.50
$I_8$	20	11	38	2	32	7	23	1	38	16.75	98	100.00	50.00
Average results									25.25	9.60	40.75	98.44	79.70