Color Structure Descriptor-Based CBIR System

PIV - Prog2

Joana Justo Guillaumet
joana.justo@estudiantat.upc.edu

Víctor Moreno Borràs
victor.moreno.borras@estudiantat.upc.edu

Abstract

In this report, we present a continuation of the study on Content-Based Image Retrieval (CBIR) systems, with a focus on using color information. Specifically, we explore the use of the Color Structure Descriptor (CSD) of MPEG-7 as a descriptor for characterizing images. We evaluate the performance of the CSD-based CBIR system and compare it to the histogram-based approach, which was previously studied in the first version of this system. We use quantitative measures such as the F-measure and the computational cost to evaluate the results obtained from the CSD-based CBIR system. Our results indicate that the CSD-based CBIR system outperforms the histogram-based approach in terms of retrieval accuracy, while incurring a slightly worst computational cost. This study highlights the potential of using color information in CBIR systems, and the effectiveness of the CSD as a descriptor for characterizing images.

I. INTRODUCTION

Building upon the previous version [5], this report presents the advancements made in the Content-Based Image Retrieval (CBIR) system[2] done in the PIV laboratory. It is important to note that the evaluation of the second version of the CBIR system continues to employ the same database of 2,000 images featuring 500 distinct objects. This ensures consistency and allows for a direct comparison between the two versions.

Building upon the foundation of the first version, the second iteration incorporates several modifications and introduces the implementation of the color descriptor. By integrating the Color Structure Descriptor (CSD) from the MPEG-7 standard, we enhance the system's ability to characterize images based on their color information. This addition complements the previously utilized histogram-based approach, enabling a more comprehensive analysis of the image content.

II. OVERALL SYSTEM DESCRIPTION

The overall system description for the second version remains the same as the first version, with the difference being the descriptors used in the system. Instead of representing the images as histograms, the second version employs the Color Structure Descriptor (CSD), which is also represented as a vector denoted as $\mathbf{h}$.

The system takes an input file, *"input.txt"*, which contains the names of the images to be analyzed. For each image specified in the input file, the system performs the following steps:

1. Loads the matrix $\mathbf{H}$ and extracts the CSD vector, denoted as $\mathbf{h}$, from the current image.

2. Calculates the distance vector, $\mathbf{d}$, between the CSD vector $\mathbf{h}$ and all CSD vectors in $\mathbf{H}$.

3. Identifies the 10 smallest values in the distance vector $\mathbf{d}$ and retains their corresponding indices to determine the 10 image names to be written to the output file, *"output.txt"*.

All the code can be found and used in our github repository *https://github.com/victhormoreno/CBIR-System*[6].

In the following section, we will discuss the key differences between the first version and the current version of the system, focusing on the extraction of the descriptor.

Color Space: HMMD

In the second version of the CBIR System, we switched from using black and white images to RGB images, which helps capture color information. To enhance the performance of the Color Structure Descriptor, we convert the image to the quantized HMMD color space [7]. The performance of CSD descriptor increases considerably while using the HMMD color space [3].

HMMD color space consists of four color components named as H-hue, M-maximum, M-minimum and D-difference. HMMD color space components are derived from the RGB values. This conversion is shown in (1) - (4) and Algorithm 1:

$$\textbf{Maximum} = max(R, G, B) \tag{1}$$

$$\textbf{Minimum} = min(R, G, B) \tag{2}$$

$$\textbf{Difference} = Maximum - Minimum \tag{3}$$

$$\textbf{Average} = \frac{(Maximum + Minimum)}{2} \tag{4}$$

Algorithm 1: Calculate Hue based on RGB values

Input: RGB values: R, G, B
Output: Hue value
if *Maximum* = *R* & $G \geq B$ **then**
  $Hue = 60 \times \frac{G-B}{Max-Min}$
else if *Max* = *R* & $G \leq B$ **then**
  $Hue = 360 + 60 \times \frac{G-B}{Max-Min}$
else if *G* = *Max* **then**
  $Hue = 60 \times \left(2.0 + \frac{B-R}{Max-Min}\right)$
else
  $Hue = 60 \times \left(4.0 + \frac{R-G}{Max-Min}\right)$
return *Hue*

The next step is quantifying the four levels of the HMMD color space by condensing them into a single level using the following table. By utilizing the provided table, we can assign a unique level to each of the HMMD color components, enabling us to quantitatively represent the HMMD color space and perform further calculations or analysis based on it. In the case of our system, there is the possibility of quantifying the image in two ways: 128 levels and 256 levels.

Subspace	256		128	
	Hue	Sum	Hue	Sum
0 [0-6)	1	32	1	16
1 [6-20)	4	8	4	4
2 [20-60)	16	4	8	4
3 [60-110)	16	4	8	4
4 [110-255]	16	4	8	4

Table 1: HMMD COLOR SPACE QUANTIZATION

In Figure 1, the image 'ukbench00000' in RGB format and its quantized form in HMMD can be observed. The RGB image displays the original color representation, while the quantized HMMD image represents the same image but with the color information condensed into the HMMD color space using the quantization levels previously defined. This transformation allows for a more compact representation of the image's color information, facilitating further analysis or processing based on the HMMD color space.

(a) RGB (b) HMMD Quantification

Figure 1: Image 'ukbench00000.jpg' from the dataset

Once we have transformed the RGB image into quantized HMMD representation, the next step is to obtain the Color Structure Descriptor (CSD).

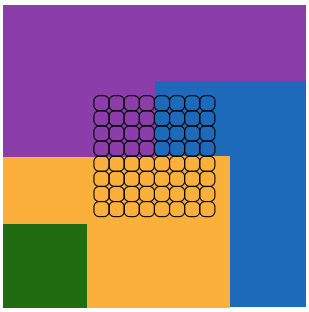
MPEG-7 Color Structure Descriptor

The Color Structure Descriptor (CSD) is a widely used technique in the field of still image retrieval. It focuses on expressing the local color structure within an image through the utilization of a structuring element, as illustrated in Figure 2. The CSD is identical in form to a color histogram, but it is different in terms of semantics. This process allows the CSD to effectively capture the distribution and correlation of colors in the image [3] [4].

CSD Algorithm

By sliding a structuring element of a certain dimension over the image pixels, the system, computes the Color Structure Descriptor. In Figure 2, a structuring element of size 8x8 is visually depicted. At each iteration of the algorithm, which corresponds to each position of the structuring element on the image, the colors of the pixels within the structuring element are examined. For each color encountered, the corresponding position in the vector **h** is incremented by one.

It is important to note that even if there are multiple pixels of the same color within the structuring element, the increment for that color occurs only once. Therefore, if the structuring element contains 64 different colors, 64 different **h** values will be incremented in that iteration. However, if all 64 pixels within the structuring element have the same color, only one increment will occur for that



Color	h(c)
c_1	$h(c_1)+1$
c_2	$h(c_2)$
c_3	$h(c_3)+1$
c_4	$h(c_4)+1$

Figure 2: Example of the CSD calculation for a 4-color image and a structuring element of dimension 8. In this case, the structuring element is centered and contains three different colors

color in that particular iteration. This process is repeated for each position of the structuring element, resulting in the complete computation of the Color Structure Descriptor, the vector **h**.

However, prior to the descriptor calculation, a subsampling must be applied to the image based on its size. Some studies indicate that the subsampling factor and structuring element size when calculating CSD follow this general rule [3] [4]:

For an image $\mathcal{I}_{h\times w}$ of dimensions, h (height) and w (width). Then, the constant p is:

$$p = \max\{0,\ \text{round}\ (0.5\cdot \log_2(h\cdot w) - 8)\} \qquad (5)$$

From p then we can compute K, the subsampling factor and E the size of the structuring element $\mathcal{E}_{E\times E}$.

$$K = 2^p \qquad E = 8\cdot K \qquad (6)$$

This rule (6) indicates that the size of the structuring element increases as the dimensions of the image do as well. The larger the structuring element E, the longer the execution time. Therefore, a downsampling factor is also applied. In our case, $p = 1, K = 2, E = 16$. The implementation of this algorithm can be found in the github repository [6]. Specifically in the **computeCSD.m** function inside the **/functions/descriptors** folder.

Once we have obtained the Color Structure Descriptor (CSD), represented by the vector **h**, we can perform a final processing step on it. Similarly to the initial histogram analysis [5], we can work with different bins. The default size of **h** will depend on the HMMD quantification we are using, either 128 or 256. From here we can unite the bins to work with a vector h with a smaller size.

III. RESULTS

Firstly, we will evaluate the system for the input file provided by the lab professor, working with 256 levels of quantification HMMD and 256 bins. To do so, we will visualize the Recall-Precision graph and calculate the F-measure. This will allow us to compare the performance of the system when using three different distance measures.

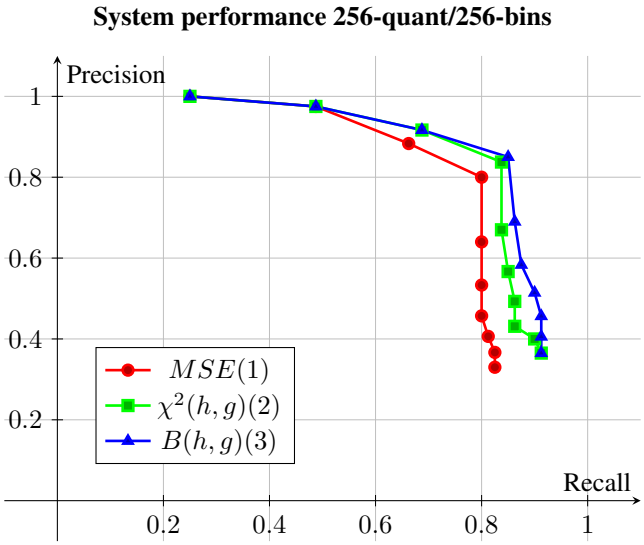


Figure 3: This precision-recall graph shows the performance of our system using three different distance metrics: Euclidean distance (red curve), Bhattacharyya distance (blue curve), and Chi-squared distance (green curve).

As well as the previous report [5], Bhatacharyya distance (3) [1] has the best performance of the three, despite using another descriptor. For this reason, from now on the study of the other characteristics of the descriptor will be done only with this distance.

	F-score	Execution time/image
128 quantification	0.825	120 ms
256 quantification	0.85	131 ms

Table 2: Results for 20 images of the input.txt

Note: Results computed with maximum bin values..

The table compares the F-score and execution time per image for two quantification methods: 128 quantification and 256 quantification. With an F-score of 0.825, the 128 quantification method demonstrates reasonably accurate classification, while the 256 quantification method achieves a slightly higher F-score of 0.85.

Bins study

Now, we will evaluate the CBIR system using different bin levels for the histograms. By comparing the results obtained with different bin levels, we can assess the impact of the bin size on the performance of the system. This evaluation will help us determine the optimal bin level that balances the level of detail in the image representation and the computational cost required for processing the images.

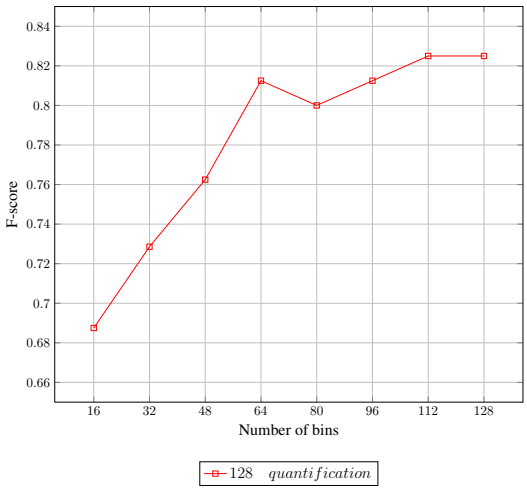


Figure 4: F-score as a function of the number of bins for Bhattacharyya distance and 128 levels of quantification.

The figures show the F-score as a function of the number of bins for quantification using Bhattacharyya distance. In Figure 4, the F-score steadily increases as the number of bins increases, reaching a maximum value of 0.825 at 128 bins.

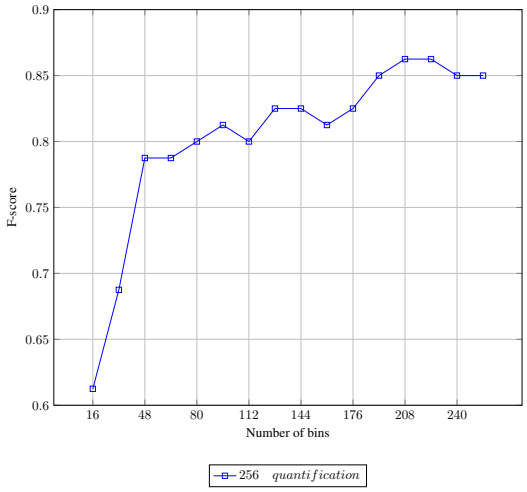


Figure 5: F-score as a function of the number of bins for Bhattacharyya distance and 256 levels of quantification.

In Figure 5, the F-score also increases with the number of bins. However, it continues to rise beyond 128 bins, reaching its peak at 208 bins with an F-score of 0.8625. Subsequently, the F-score slightly decreases at 240 bins back to 0.85.

As a conclusion, the optimal values for quantification using 256 bins are 208 and 224 bins, as they yield the highest F-scores. However, considering computational cost, it is more favorable to choose 208 bins since it achieves a slightly lower F-score but with lower computational requirements.

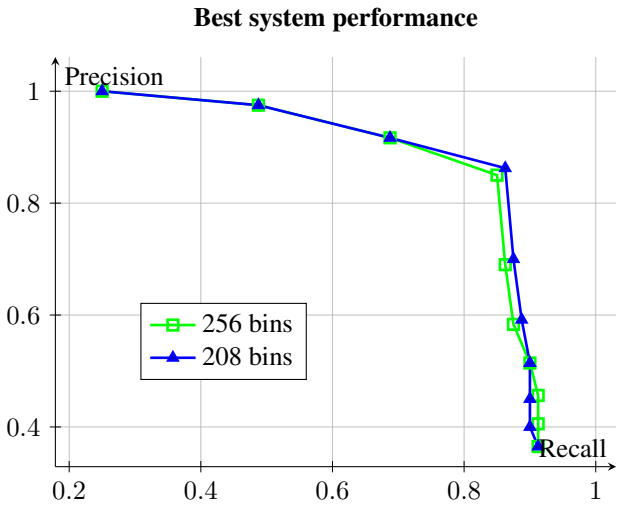


Figure 6: In green the precision-recall graph of the system using the Bhatacharyya distance, 256 quantization levels and the maximum of 256 bins. In blue the same but using 208 bins instead of the maximum.

In Figure 6 we can see the improvement of using 208 bins compared to the best system configuration without studying the bins. Besides better in terms of system performance, from 0.85 to 0.8625 of F-Score, by using a smaller amount of bins we improve both the execution time and the memory used to store the database.

We have made the decision not to display the runtime per image based on the number of bins, nor consider it as a criterion. This choice is due to the observation that there was minimal change in execution times based on the number of bins. Furthermore, during our study, we noticed that the runtime was highly sensitive to the conditions under which the study was conducted and did not yield consistent results. As a result, we believe it is more reliable to focus on other metrics that provide more meaningful insights.

IV. CONCLUSIONS

It is worth noting that the results presented here are based on a specific experiment, specifically the images of the input.txt, and may not be generalizable to other contexts. Therefore, the results should be interpreted with caution and further experimentation with others input.txt may be needed to confirm the conclusions.

In general, the results obtained align with our expectations. We observed that, unlike the first version [5], the study of bins has exhibited a suitable behavior, indicating that a higher number of bins leads to better performance. However, it is worth noting that there was a deviation from this trend in the case of 208 bins.

The optimal configuration for our for this specific input file appears to be using the Bhatacharyya distance for 208 bins with 256 levels of quantification. Below, in Table 3, you can observe the best configurations of the system, based on its performance and the number of bins used, listed in order from best to worst.

Descriptor	Bins	Distance	F-score	Time/imag
CSD 256-quant	208	Bhatacharyya	0.8625	130 ms
CSD 256-quant	192	Bhatacharyya	0.85	128 ms
CSD 128-quant	112	Bhatacharyya	0.825	120 ms
CSD 128-quant	128	Bhatacharyya	0.825	120 ms

Table 3: Better system settings

In conclusion, for optimal performance with a higher number of bins, it is recommended to use 208 bins instead of 256 bins. The configuration with 208 bins achieved a higher F-score of 0.8625 while using less memory. Alternatively, if low memory usage is crucial and a lower number of bins is required, it is advisable to choose 112 bins. However, the optimal configuration for the system is based on a specific experiment and may not be generalizable to other contexts. Further experimentation with other input files may be necessary to confirm the conclusions.

Improvements

We can divide the potential improvements of the system into two categories: enhancing the way the image is described and improving the decision-making process. For the first part, alternative techniques such as GoP/GoF (Group of Frames or Group of Pictures Descriptor) can be employed to yield better results. In the second stage, a possible improvement could involve utilizing a machine learning system as a decision maker and training it using the descriptors obtained. This approach would allow the system to learn from the descriptors and make more accurate decisions.

Additionally, another approach to consider is replacing the entire system of descriptor and decision-maker with deep learning. This could involve employing a convolutional neural network (CNN) that directly handles both stages consecutively. By training the CNN on a large dataset, it can learn to extract meaningful features from the images and make decisions based on those features. we have been able to verify the good behavior of these types of system in the laboratory's practical number 6, which demonstrates the promising of this end-to-end deep learning approach across diverse image classification tasks.

References

[1] Authorless. Bhattacharyya distance. `https://en.wikipedia.org/wiki/Bhattacharyya_distance`.

[2] Authorless. Content based image retrieval. `https://en.wikipedia.org/wiki/Content-based_image_retrieval`.

[3] Abdessalem Ben Abdelali, M.N. Krifa, Lamjed Touil, Mtibaa Abdellatif, and El-Bay Bourennane. Astudy of the color structure descriptor for shot boundary detection. *International Journal of Sciences and Techniques of Automatic control computer engineering, IJ-STA*, 3:956971, 06 2009.

[4] Adis Buturovic. Mpeg 7 color structure descriptor for visual information retrieval project vizir. *Institute for Software Technology and Interactive Systems Technical University Vienna*, 2005.

[5] Víctor Moreno Joana Justo. Content based image retrieval (cbir) - histogram based descriptors, 2023.

[6] Víctor Moreno Joana Justo. Piv-prog-2. `https://github.com/victhormoreno/CBIR-System`, 2023.

[7] L. K. Pavithra and T. Sree Sharmila. Image retrieval based on chrominance feature of the hmmd color space. In *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*, pages 1–6, 2017.