Application of Calculating Similarity of Portrait Paintings

| Group Member |
| --- |
| Julio Nevado Delgado |
| Wenqi Jiang |
| David Cabornero Pascual |

# **Introduction and Dataset**

As technology evolves, an ever-increasing number of images arise on the Web, demanding the extraction of information from these images are very urgent in many different industries such as data mining, personalized recommendations, advertising services etc, so more and more algorithms and techniques are being created to meet these demands.

The situation is similar in the world of art, where the centuries-long history of contemporary human civilization has produced a vast number of magnificent works in a variety of classical but aesthetically distinctive styles. This application used different content-based image retrieval techniques (CBIR) to calculate the similarity of portraits in different paintings. It would be quite beneficial to the art practitioner or museum manager in easily discovering relevant artworks.

WikiArt, a well-organized enormous database with more than 80,000 fine-art paintings in 27 various creative genres, ranging from the fifteenth century to present times, is the major data source for those portrait paintings. We used the images pre-processed by Jiaqi Yang[1] directly, and these pre-processing steps are as follows:

1. Filter out all the portrait paintings from WikiArt dataset
2. Choose the six art movements
3. Face-crop code is applied to auto generate face images
4. For failed auto face cropping, we manually crop them

After processing, the 927 portraits were finally extracted from the 923 original paintings. After pre-processing these images and analyzing them using methods such as Color Histogram and Wavelet-Based Color Histogram(WBCH), we finally calculated the similarity between the portraits and ranked them once a query is given.

# **State of art: CBIR Algorithm**

In the study carried out by Manimala Singha and K.Hemachandran [2], they tried to calculate a distance metric between two different images. They divided this job into two different metrics: Color Histogram and Wavelet-Based Color Histogram (WBCH).

**Color histogram** algorithm creates a histogram that counts the number of pixels with a certain color. Each bin represents a certain range of colors and represents how often this range appears in the image.

In particular, we start with an RGB (red-green-blue) division of the image that we have to transform into an HSV one (hue-saturation-value), which usually works more properly in this kind of task. The next step is to divide the hue, the saturation and the value respectively in 8 equal parts. With this, every pixel can be defined by a 3-dimensional vector whose values are between 0 and 7. Finally, we create a 8x8x8 matrix where we store how many pixels are in each range of values.

**Wavelet-Based Color Histogram** creates a histogram again, but in this case this algorithm is more aware about the texture property of the image. Taking into account that we are working with art pieces, the texture could be more relevant if we are trying to assert that the distance between two pieces that belong to the same artistic movement should be lower.

Now, we start again with an RGB decomposition of the image, but now we will decompose each of the colors with the Haar Wavelet transform. This transform divides each color in an approximate coefficient, and vertical, horizontal and diagonal detail coefficients. After that, we have to merge again the red, green and blue components, obtaining three images that we are going to use: the approximate one, the horizontal detail and the vertical detail. FInally, these three images are merged with these respective weights, 60%, 20% and 20%. This final RGB image is converted to a HSV image, and following the previous method in the color histogram (creating a 8x8x8 histogram) we obtain the final result.

Now, we have two matrices associated with each painting. The **distance** between two art pieces is calculated as follows:

1. Color Histograms are compared by a norm of the difference of the matrices. The same distance is considered for WBCH.
2. Both distances are averaged in order to get the final distances.

With this, we can define a distance between each pair of art pieces and create a ranking of the most similar paintings on a dataset regarding a certain query (another painting).

# **CBIR Implementation**

## Histograms

The implementation proposed involves the computation of the color and texture histograms of the query image as well as the computation of both histograms for each other image in the database. Both histograms are then combined in a simple way using the average. In order to obtain a histogram, images are first transformed to HSV. HSV represents a color space which is defined by hue, saturation and value. In the case of the color histogram, this transformation is made directly over the loaded image in BGR format (the default format in CV2) while in the case of the texture histogram the input image undergoes some other transforms (specified in the state-of-the-art section) before being converted to a HSV image.

As said in the state-of-the-art section, histogram computation yields a matrix of dimensions 8x8x8. In that three-dimensional matrix, the coefficients are interpreted as follows. Supposing we have a coefficient of 12 in the position of the matrix 3x3x2, the meaning is that in the HSV formatted image there exist 12 pixels that have a hue of 3, a saturation of 3 and a value of 2.

The following functions from CV2 are used for the implementation:

* calcHist: computes the histogram based on an HSV image.
* cvtColor: allow us to transform images between different color spaces.
* split: allow us to split image components (from a BGR image obtain three matrices, B, G and R).
* merge: the opposite to split, takes several matrices and creates a new image by combining them.
* addWeighted: combines images by weighting them.

## Distance metric

For the distance metric, the histogram intersection described in [2] has been implemented:

However it is an intersection, so the higher the more similar images are. To solve this we considered as distance .

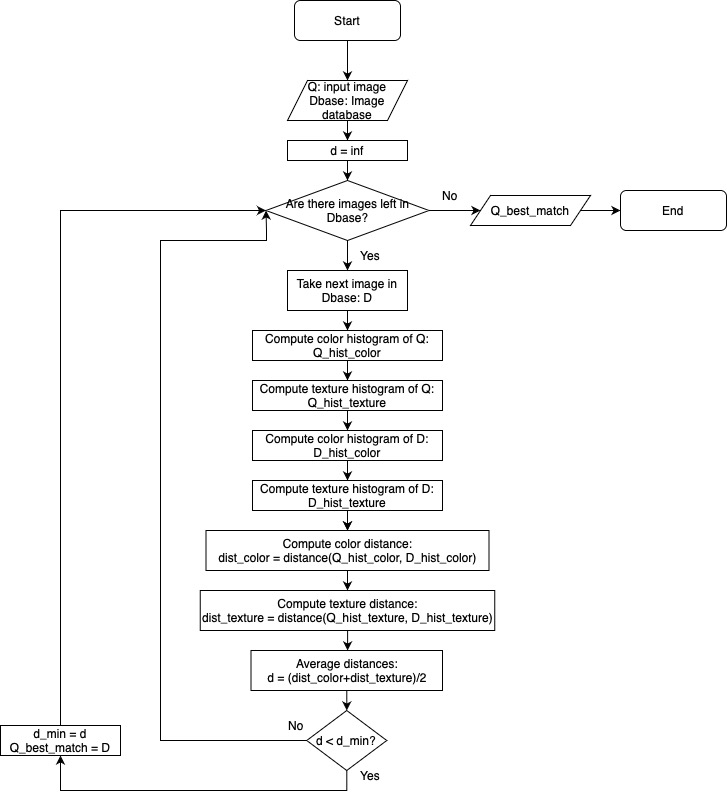
Q and D are the flattened histograms of the query image and an arbitrary image in the database. Histograms are flattened to ease comparison since we are not interested in knowing that 12 pixels have a hue of 3, a saturation of 3 and a value of 2 but in knowing that there exists 12 pixels in the query image (for instance) that have a certain combination of hue, saturation and value. Both histograms are flattened the same way so each position in each array of the query and database image matches, represent the same hue, saturation and value settings.

Also distance was implemented attending at the following formula:

Where Q(i) and D(i) keep the same meaning as in the previous formula. Distance will be selected between intersection-based and based on user input.

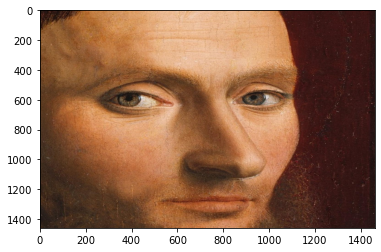
We will iterate over every image D in the database (except for Q) and compute the distances (using the selected one) between both, color and texture histograms. Final distance is obtained by averaging both of them. Finally, the image with the smallest distance is obtained.

For the sake of clarity, a flowchart with the code logic is included.



# Conclusions

In this research, we developed an application for calculating the similarity of portrait paintings and identifying the portrait with the highest similarity among thousands of pictures. Here is an example of our results.



# 

Figure 1: On the top left, the query image. On the top right, the most similar one according to the intersection distance. Below, the most similar one according to the chi-squared distance. They belong to the same artistic current: renaissance.

We implemented a new method for Content Based Image Retrieval by combining the color and texture features which is called Wavelet-Based Color Histogram Image Retrieval (WBCHIR), and a distance function to determine how similar these artworks are. Furthermore, our application substantially reduces the number of processing steps. As a consequence, retrieval speed has increased dramatically. The whole process just took 30 seconds for all 927 portrait paintings.

# References

[1] Yang, J., 2022. *Portrait Painting Dataset For Different Movements*. [online] Mendeley Data. Available at: <https://data.mendeley.com/datasets/289kxpnp57/1> [Accessed 15 March 2022].

[2]M. Singha, "Content Based Image Retrieval using Color and Texture", *Signal & Image Processing : An International Journal*, vol. 3, no. 1, pp. 39-57, 2012. Available: 10.5121/sipij.2012.3104.

# **Appendix: Work assessment**

| **Task** | **Status** | **What** |
| --- | --- | --- |
| Proposal of CBIR application in a real problem | Implemented | Paintings from different eras continue to be discovered. CBIR allows us to match them based on already cataloged ones in order to frame them into a well-known artistic movement. |
| Technical proposal of toy CBIR applied to a real problem based in Literature | Implemented | We follow Manimala Singha’s proposal. Described in the state-of-the-art section. |
| Implementation of the ‘smart’ Histogram descriptor | Implemented | We implement a color histogram. |
| Implementation of the ‘smart’ Histogram distance | Implemented | We implement a distance based on histogram intersection. |
| Implementation of a secondary descriptor and distance | Implemented | We implement a texture based histogram. We also implement the distance. |
| Implementation of CBIR processes (store the indexes and search) | Implemented | User introduces filename path. Image is opened and histogram computed. Histograms for each image in the database are computed and compared. Most similar image is retrieved. |
| Documentation | Implemented | Present report + code in the same zip file. |
| Presentation | Implemented | Presentation in the same zip file. |

# 