Museum Query Retrieval with Classic Computer Vision

Alex Carrillo[1] Alvaro Budria[1] Irene Estévez[1] Johnny Nuñez[1]
[1] *Universitat Autònoma de Barcelona*

Abstract. This technical report presents a summary of the work developed during the M1 module ("M1. Introduction to Human and Computer Vision"). The aim of this project was to learn the basic computer vision concepts and techniques to build an image retrieval system for finding paintings in a museum image collection. The image retrieval was performed based on several image descriptors, such as color-based, texture-based, text information, key-points and local descriptors. At the retrieval stage, the comparison between a query descriptor and the database descriptors was performed by similarity measures. Some other techniques were also applied, for example different types of filters, for detecting and removing unwanted background, noise or text-boxes from the studied images. The viability of the studied methods and techniques is demonstrated experimentally.

Keywords: Keywords: Image Preprocessing, Feature Extraction, Text Extraction, Query Retrieval, Computer Vision in Art

1 Introduction

In recent years, the field of computer vision has grown with many advances in algorithms and hardware. Computer vision is nowadays used in a wide range of applications such as biomedical imaging, commerce, autonomous vehicles, Internet, among others. This discipline is able to obtain information from raw images automatically after suitable processing. How is possible to obtain information from images? Along this paper, we show a set of different methods that allows us to identify a particular image on a database (DDBB). In this manuscript, we describe how to build an image retrieval system for finding paintings in a museum image collection based on several types of well-known features, namely color-based, texture-based, text information and keypoints descriptors. Note that some sophisticated and complex algorithms were considered to extract different features of the query images to complete our objective, i.e., to obtain specific characteristics of the query image to predict and recommend images of the museum DDBB.

By taking into account that some of the studied query images contain noise, several pictures per image, undesired background, etc., several preprocessing techniques are also implemented. Before extracting the features and computing similarities, the images under study were preprocessed: the query images were

denoised, a mask was applied to detect the number of paintings per image and the textbox positions, and subsequently to crop them. In addition, rotated images were corrected. We want to remark that preprocessing is one of the most critical steps of images analysis.

The structure of this paper is the following: In Sec. 2, the studied preprocessing techniques are summarized. Sec. 3 provides a brief description of the feature extraction step. Sec. 4 introduces our image query and image retrieval system. In Sec. 5, the most relevant results are presented. And finally, Sec. 6 discusses conclusions and future steps.

2 Image Preprocessing

The aim of preprocessing is to enhance the quality of query images, avoiding undesired image transformations, such as rotations, noise, unwanted background, etc. This process results in an accuracy increase in the downstream retrieval task.

2.1 Denoising

Denoising images is the process of removing noise from images. This process can help improve the quality of images, and make them look sharper and more clear. There are different types of noise such as Gaussian noise, salt and pepper, shot noise and impulse noise [2].

One way to detect noise in images is to look for patterns of pixels that are different from the surrounding pixels. This can be done by looking at the image histogram, or using a noise detection algorithm. The *Estimate sigma* [6] algorithm is a reliable choice, as it is designed to be robust to different types of noise of varying intensities. We use it to estimate a σ for each image in the dataset. Then, a threshold on σ is assigned to decide whether an image is considered as noisy or not.

There are many methods for denoising images, including using image editing software, or applying filters to images. After experimenting with **Gaussian**, **mean** and **median** filtering, and **wavelet denoising**, we settle with the median filtering, as it is the most robust to different kinds of noise.

2.2 Background Removal

Flood Fill is a color-based, region growing segmentation algorithm that progressively «fills» or segments pixels having a color similar to that of the initial color the process starts from [9].

In our implementation, as the paintings in the multiple query images with background are more or less centered, we assumed that the upper left corner pixel belongs to the background. Therefore, this approach is a suitable starting point. In practice, neighbor pixels are considered as having the same color within a certain tolerance, which we set to ± 4 in 255-valued (8-bit) images.

Otsu Binarization Otsu's Binarization method [8] only takes into account the color histogram of the image. In the case of binary segmentation we are dealing with, the threshold is found by iteratively increasing the candidate threshold throughout all the intensity range, and selecting the one resulting in the greatest intra-class variance.

Morphology-based Segmentation Morphology is a set of image processing operations based on gray-scale shapes that apply a structuring element to an image. The value of each pixel in the output image is a comparison of the given pixel in the input with its neighbors.

Multiple Painting Segmentation The segmentation methods discussed above generate a single mask image. When the image contains several paintings, as may be the case in some of the query data-sets, the computed mask should also contain several disconnected regions corresponding to each one of the paintings. Therefore, it is necessary to extract each of the masks separately as an additional step, to be able to deal with images having multiple paintings.

In this way, we find all connected components within the given mask, and keep the n biggest ones having at least 5% of the total area, to avoid false positives. Here n is the maximum number of paintings that images are expected to contain, and varies with the data-set. Once connected components are found, they are filled. The resulting mask can be used to crop the corresponding image containing several pictures.

2.3 Textbox Detection and Removal

In order to place a bounding box representing the rectangular areas of images containing text (and generate their corresponding masks), the morphological gradient is a good choice. Text can be well discriminated from other high-gradient zones because it is placed over uniform (black or white) boxes.

The resulting image shows the areas where the gradient is close to 0, and therefore, which ones can be removed using either an opening or a closing. We used the gradient of an opening or a closing operators because it is invariant to the intensity differences that might appear between them (it is more robust than simply using the operators by themselves).

2.4 Image Rotation

To estimate the bounding boxes coordinates and the rotation angle of paintings, we use the **Principal Component Analysis (PCA)** statistical procedure to extracts the most important features of a given data set and get each painting angle. Note that in the case of images with multiple paintings, we rotated the entire image by the mean predicted angles of each painting.

3 Feature Extraction

A critical part of our query retrieval pipeline is the feature extraction, whose goal is to project images to a feature space in which they can be compared against each other. We employ several kinds of features descriptors, which we categorise as color-based, texture-based, text-based and keypoint-based descriptors.

3.1 Color-based Descriptors

Descriptors falling under this umbrella use the color distribution of images to extract meaningful information from them.

Color Spaces Several choices for color space are possible when it comes to represent an image. For this work, we used the color spaces shown in Fig. 1, i.e. Red-Green-Blue (**RGB**), **Grayscale**, **CieLAB**, Hue-Saturation-Value (**HSV**) and **YCrCb**.

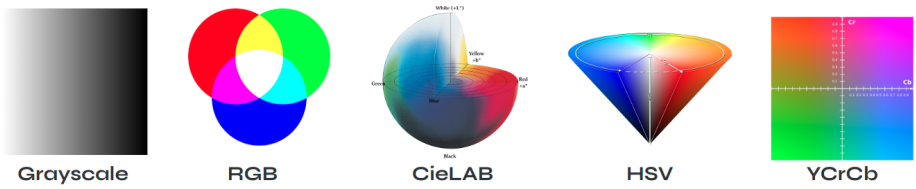

Fig. 1. Graphical representation of the color spaces we experiment with.

For this color spaces, we computed different color descriptors: 1D histograms, 3D histograms and multiresolution block-based histograms.

1D Histogram. A 1D histogram is a per-channel representation of an image that counts the amount of times each channel value appears within the image, regardless of its position:

$$h_c(i) = \sum_{i=1}^{D_{max}} \frac{n_i}{N}$$

where c is the channel index, D_{max} the maximum display value, i a pixel value, n_i the amount of pixels having this value, and N the total amount of pixels.

In our implementation, we used normalized 1D histograms that add up to 1. If a multichannel color space is used, we end up with a histogram per channel, so we concatenate all histograms into a single one. We also quantized the histograms into a number of bins $n_{bins} < D_{max}$ to reduce the feature space dimension and avoid the curse of dimensionality.

3D Histogram. In a similar way, we computed 3D histograms by counting the amount of times each tuple of channel values (i.e. (R_i, G_i, B_i)) appears within a given image. Unlike in the case of the 1D histogram, the result is a 3-dimensional block. To obtain a final vector representation, we flatten the 3D histograms.

Multiresolution block-based histogram. The naive histogram explained above has the downside that it totally ignores contextual information. To alleviate this issue, we extracted histograms at several scales or levels L. Moreover, to incorporate locality into the representation, we divided the image into $2^{(L-1)} \times 2^{(L-1)}$ patches at each level L (see Fig. 2). The histograms obtained at each patch were normalized and concatenated into a single vector.

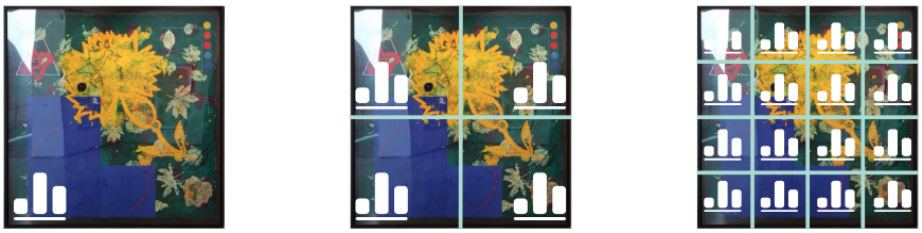

Fig. 2. In the multiresolutin block-based histogram, the image is divided into $2^{(L-1)} \times 2^{(L-1)}$ patches at each level L.

3.2 Texture-based Descriptors

Descriptors based on texture use local information about gradients, edges and frequencies to derive features. Although many options exist, in this work we focused on the three most popular features: DCT, LBP and HOG.

Discrete Cosine Transform (DCT) The Discrete Cosine Transform (DCT) [1] expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. To extract localised information, each image is partitioned into a collection of 8×8 sized blocks, and each of these blocks is decomposed as a weighted sum of the DCT basis vectors. The resulting coefficients are projected onto a lower dimensional space using Principal Component Analysis (PCA), after which they are ready to be used for comparing samples (see Fig. 3).

Local Binary Pattern (LBP). Local Binary Pattern [10] is a texture-based visual descriptor. This local descriptor uses the relative change illumination between each pixels and its surroundings. Therefore, it is invariant to photometric changes and changes in illumination.

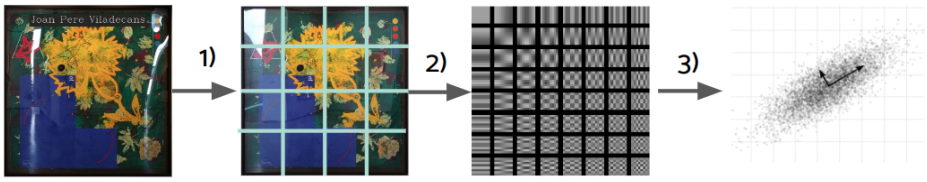

Fig. 3. Feature extraction pipeline with DCT. 1) Partition the image in 8×8 sized blocks. 2) Each block is projected onto the DCT basis. 3) The DCT coefficients are projected onto a much lower dimensional space.

Some of its hyperparameters include the radius neighborhood (R), points used for computing local feature (P), and number of bins in histogram of patterns (N_b). In our experiments, we found $R = 8$, $P = 4$, and $N_b = 16$ to yield good results.

Histogram of Oriented Gradients (HOG). The Histogram of Oriented Gradients [5] feature descriptor counts occurrences of gradient orientation in localized portions of an image. It is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

The main hyperparameters are the number of orientations, the number of pixels per cell, and the number of cells per block. Typical values for these hyperparameters are 8 orientations, 16^2 pixels per cell, and 1 cell per block

3.3 Text: Optical Character Recognition (OCR)

In order to detect the text of some paintings, we performed Optical Character Recognition (OCR) by using only the rectangular areas of images containing text (via the bounding boxes placement). In this case, we use the Python `pytesseract`[1] package with a **catalan** language model of `tessdata_best` (Tesseract's best and more accurate trained models) from the original repository[2].

3.4 Keypoint-based Descriptors

The last family of descriptors we experiment with are keypoint-based descriptors, which first localize a relevant subset of points within the image (e.g. distinctive corners) and then return a local descriptor for each of the keypoints. A query image is matched to a database image individually comparing each feature and finding candidate matching features based on the distance between their feature vectors. Further details on matching keypoint-based image representations are given in Sec.4.3.

[1] Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images.

[2] https://github.com/tesseract-ocr/tessdata_best

Scale Invariant Feature Transform (SIFT). The Scale Invariant Feature Transform (SIFT) [7] method for image feature generation transforms an image into a large collection of feature vectors, each of which is invariant to image translation, scaling, and rotation, partially invariant to illumination changes, and robust to local geometric distortion. Key locations are defined as maxima and minima of the result of Difference of Gaussians (DoG) function applied in scale space to a series of smoothed and resampled images.

SIFT presents several hyperparameters: number of octave layers N_L in DoG, contrast threshold T_c for filtering out low-contrast regions, edge threshold T_e for filtering out edge-like features, and scale σ of Gaussian applied at the beginning of the pipeline. In our experiments, we set $N_L = 3$, $T_c = 0.04$, $T_e = 10$ and $\sigma = 1.6$.

Oriented FAST and Rotated BRIEF (ORB). Oriented FAST and Rotated BRIEF (ORB) [12] is a faster lightweight alternative to SIFT. It builds on FAST [11] and BRIEF [3] methods to produce rotation and scale invariant keypoint features from images. Thus, ORB presents higher expressiveness while being less computationally expensive to compute than SIFT.

The hyperparameters in ORB and their values in our experiments are: the pyramid decimation factor $F_P = 1.2$, the number of pyramid levels $N_L = 8$, the edge threshold $T_E = 31$ for filtering out edge-like features (the smaller this value, the less features are produced), and the size $S_P = 31$ of the patch utilized by the oriented BRIEF descriptor.

Accelerated-KAZE (AKAZE). AKAZE algorithm is based on non-linear diffusion filtering, which formulates stable non-linear scale space to resolve the problem of boundaries and detail loss. Despite its theoretical advantages, AKAZE is the most computationally expensive feature extraction algorithm out of all the ones we have presented. This fact stems from the slow computation of the diffusion.

4 Image Query and Retrieval

Once descriptors have been obtained for each of the query and database images, it is necessary to match query images in the data-set to their corresponding ones in the database. To do so, a measure of similarity between each query image and all images in the database is computed. Then, database images for each query can be ranked based on their similarities. In this section, we present how the similarity measures are computed depending on the descriptor type we are dealing with.

4.1 Feature Similarity

Most of the descriptors used in this work full under the category of generic features. In this case, we assume a feature vector for each query image and

database image. The computed set of feature vectors per query image is compared against each other. For this work, we experimented with several different distance measures, namely, L^1 and L^2 norms, χ^2 distance, **Hellinger kernel**, **cosine similarity**, and **histogram correlation** (when applicable). Unless otherwise specified, in our experiments we default to Hellinger kernel as similarity measure.

4.2 Text Similarity

An image may be described based on text or a string of characters that have been extracted from it. In this case, we compared paintings through their corresponding text. Three different distance measures were employed during the experimental work: **Hamming distance**, **Levenshtein** distance, and **Damerau-Levenshtein** distance.

4.3 Keypoint Matching

In the particular case of keypoint-based descriptors, a somewhat more involved process must be followed to compute similarities between paintings. The overall pipeline for matching paintings with keypoint descriptors is as follows:

1. Compute keypoint descriptors for each image (of both the query data-set and database).
2. Match keypoints with either a brute-force (BF) matcher (in which case a similarity measure must be chosen), or with a fast nearest neighbors approximation.
3. Filter out false positives either by crosschecking the candidate keypoints, by applying Lowe's ratio test, or both. Crosschecking returns only those matches that are corresponded. Lowe's ratio test first picks the two best matches with lowest distance to the given keypoint. Then it checks whether the two distances are sufficiently far apart.
4. To retrieve from the database the most similar images to a given query image, the database images are ranked based on the number of matches with the query. If no database candidate obtains a number of matches above a certain threshold, the query is considered to have no corresponding paintings in the database.

4.4 Painting Clustering

With the aim of producing a selection of paintings and arranging them in an art presentation, we cluster the database paintings in feature space, with two different unsupervised clustering methods, KMeans and hierarchical clustering, with texture descriptors and keypoint descriptors, respectively.

Texture-Based Features and KMeans. As a texture based descriptor, we employed histogram of oriented gradients (HOG) [5] due to its excellent performance and popularity. HOG produces a very high-dimensional feature space. To avoid the curse of dimensionality, we project the features on a lower-dimensional space with PCA. Finally, we clustered the paintings into 5 different groups with KMeans.

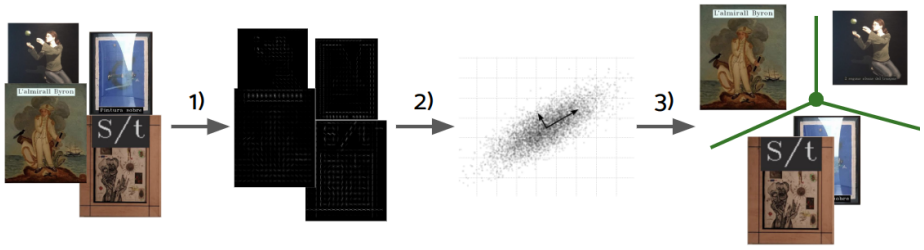

Fig. 4. Clustering pipeline with texture-based descriptors. 1) Extract HOG features. 2) Project onto low dimensional space with PCA. 3) Cluster with KMeans.

Keypoint-Based Clustering. As keypoint-based feature extractor, we choose ORB. We compared each image against all others, computing a set of matches in each comparison. From this, we obtained a square affinity matrix relating the database images, where the affinity for an image pair is given by the number of matches between the two.

For clustering paintings together, we use hierarchical clustering with complete linkage. Since this technique requires a distance matrix, we transform the affinity matrix described above into a distance one.

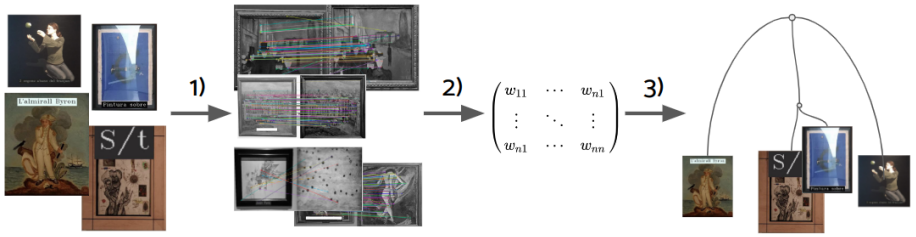

Fig. 5. Clustering pipeline with texture-based descriptors. 1) Extract ORB keypoints and features. 2) Match keypoints and obtain affinities between them. Convert affinities to distances. 3) Cluster with hierarchical clustering.

5 Results

During the development of this project, an extensive experimental study was carried out in order to analyze the effectiveness of the aforementioned methods and techniques. In this section, only the most relevant results are presented. In Fig. 6, an overview of the pipeline followed for Museum Painting Retrieval is shown. As can be observed, the proposed pipeline has two major phases: i) a pre-processing phase (labeled as denoised and rotation crop in Fig. 6) and ii) a query phase (labeled as features, feature space distance and clustering in Fig. 6).

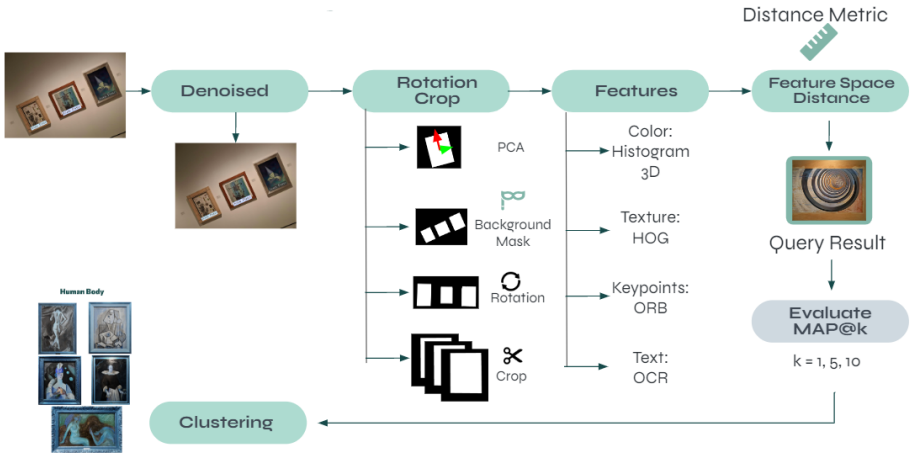

Fig. 6. Illustration of the proposed pipeline.

5.1 Data

In this subsection, a brief description of the studied images is presented. The proposed image retrieval systems were evaluated each week with a different query image data-set that consisted of 30 images some of them containing: undesired noise, superimposed text, color changes, multiple queries (1, 2 or 3 paintings) per image, rotated paintings and pictures not contained in the database (see Fig. 7) to evaluate the performance of our approach in different scenarios.

Moreover, our database (DDBB) contains 287 images of original paintings from Can Framis, Figueres and Kode Bergen Art museums. Note that each image of the query data-set and database has a different number of pixels. For this reason, before computing the feature vectors and the similarities, all the pictures were scaled, obtaining 500×500 resolution images.

Fig. 7. Set of images from the QSD1-W5 query data-set (week 5) containing noise, superimposed text, rotated pictures and multiple queries.

5.2 Noise removal

As stated above, image noise can become a major issue for feature extraction and background removal. To solve this problem, the pre-processing procedure described in Sec. 2.1 was applied to enhance images by removing the noise of some query images (see Fig. 7). We applied all the corresponding filters summarized in Sec. 2.1 for each image of the query data-set, obtaining the highest PSNR value and at the same time reducing the sigma value, since a small sigma value loses image information. Therefore, we calculated the difference between the new denoised image and the original image, thus maximizing the image energy. The PSNR and the new sigma were calculated from this difference (see Fig. 8)

Fig. 8. Example of two types of noise filters: Gaussian filter (left image) and Median Filter (right image).

5.3 Painting extraction

On the one hand, background removal algorithm has proven crucial to improve our results. In particular, if the cropped painting contains noise or misses information, the image retrieval system performance degrades. On the other hand, the rotation on paintings had no effect on the performance of most descriptors. For instance, the color-based or keypoints-based systems are rotation invariant. Nevertheless, rotating the pictures (parallel to the horizontal axis) helped us in tasks like text extraction.

Background Extraction and Segmentation For the morphology-based segmentation, we leveraged the Canny edge detector [4]. After detecting edges, we applied a *closing* with a large structuring element (50×50) to fill holes, and a *dilation* with a small one (5×5) to connect edges. Finally, we used a *reconstruction by erosion* with a 1-pixel-padded frame of the image as the marker to get the final mask.

As stated in section 2.2, three different methods were studied during Week 1 for background extraction. The obtained results for these methods are summarized in Table 1. The Otsu Binarization method was the one presenting the best results.

Table 1. Results for the best studied background removal methods.

Studied method	**Color space**	**Precision**	**Recall**
Mean	CiebLab	0.92	0.90
Otsu	Grayscale	**0.93**	**0.94**
Flood fill	RGB	0.79	0.97

Image rotation. To estimate the bounding boxes coordinates and the rotation angle of paintings, we used the **Principal Component Analysis (PCA)** statistical procedure to extract the dominant direction within a given image. We applied PCA to a predicted mask of an image (a white rectangle on a black background) to find the direction along which the data varied the most (principal components). Then, we computed the furthest left, right, top and bottom points on the PCA basis, and projected these points back to the original basis (image) to get the four corner coordinates of the painting.

With this information in hand, we computed the image rotation angle by using the two lower corners predicted for the painting, and rotating both the original images and the predicted masks, if needed. Note that in the case of images with multiple paintings, we rotated the entire image by the mean predicted angles of each painting.

The angular error for image rotation is shown in 2.

Table 2. Results of angular rotation for QSD1-W5 data-set.

	Mean	**Median**	**Std**
Angular error	5.28°	5.28°	5.28°

In addition to being rotated, the number of pictures (1, 2 or 3) on each query image were detected and masks were used to separate and crop the images of the query data-set.

5.4 Text Detection and Extraction

Errors on the textbox extraction are harmful for the text-based query system. As can be observed in Fig. 7, there is a difficulty since the text can be lighter or darker than its respective containing box, an *opening* or *closing* using a rectangular structuring element (10×30) will remove the text and keep its background. Thus, by using the method described in Sec. 2.3, we computed both operators, calculated their morphological gradients and selected the minimum from both.

The resulting image showed the areas where the gradient was close to 0, and therefore the ones that could be removed using either an opening or a closing. Finally, we masked the original image gradient with this intermediate image (the one with values close to 0), and computed a threshold ($T = 0.8 \cdot \max(\cdot)$) to filter parts of this gradient, revealing the text region.

The evaluation of text boxes detection was done by using the bounding boxes coordinates and computing the mean **Intersection over Union (IoU)** metric. Considering the set of 30 images of QSD1-W2, we obtained a mean IoU of 54.91%.

Table 3. Evaluation of text distances based on different preprocessings for QSD1-W2.

Preprocessing	**Distance**	**Mean**	**Median**
None (RGB image)	Hamming	37.63	38
	Levenshtein	26.57	24
	Damerau-Levenshtein	26.57	24
Greys image	Hamming	**37.47**	38
	Levenshtein	**25.77**	24
	Damerau-Levenshtein	**25.77**	24
Otsu binarized image	Hamming	37.70	38
	Levenshtein	25.77	24
	Damerau-Levenshtein	25.77	24

As stated in Sec. 3.3, we used Python-tesseract to extract the text of some paintings. Moreover, we also filtered out any characters that do not appear in

the database, to query from textual painting and artist names. Table 3 shows a comparison of the distances between the predicted text and the ground truth text, for QSD1-W2 (lower values mean more similarity). Note that using the generated bounding boxes and preprocessing them to a grayscale image performs better for OCR across our edit-based similarities.

5.5 Query Retrieval

In this section, we evaluate the image retrieval system for the descriptors proposed in Sec. 3. As stated above, the similarity was measured between the feature vector of the previously filtered and cropped query painting and the feature vectors of the paintings in the database. The system returns the DDBB images that were most similar to the query image. For evaluating the performance of our system, the mean Average Precision at k (MAP@k) metric was used.

Table 4 shows the MAP@1 and MAP@5 for QSD1-W5 query dataset for different types of features. The results shown in Table 4 are for:

- Text-based descriptors: Grayscale color space for detecting the text areas and Optical Character Recognition (OCR) for extrating the text. The similarity between the texts were computed by the Damerau-Levenshtein distance metric.
- Color-based descriptors: Multiresolution block-based 3D histogram (three levels are computed, normalized and then concatenated for a bin size of six and a number of blocks of $2^{level} - 1$) for RGB color space. The similarity between the features vectors of two images were computed by the Hellinger distance metric.
- Texture-based descriptors: Histogram of Oriented Gradients (HOG) and similarities computed by the Hellinger distance metric.
- Keypoints descriptors: Oriented FAST and Rotated BRIEF (ORB) with match method brute force (as it presents the best trade-off between accuracy and compute time, see Table 5), a Lowe's ratio of 0.7, a matches threshold of 20 (see Table 6) and Hamming2 distance metric.

Table 4. Best Map@k ($k = 1$ and 5) results (in percent) for different descriptors for QSD1-W5 data-set

Descriptors	**MAP@**1	**MAP@**5
Text descriptors	17.07%	25.77%
Color-based descriptors	36.59%	38.01%
Texture-based descriptors	31.70%	32.93%
Keypoints descriptors	90.24%	93.50%

Table 5. Map@k (k = 1 and 5) score and computational time for each of the three studied descriptor types, with discarding threshold set to 0. Although AKAZE+BF+Hamming2 is the best combination in terms of MAP with a slight margin, we settle for ORB+BF+Hamming2 as it is significantly faster and yields almost the same precision. Tested on: Macbook Pro, 32gb LPDDR5 M1 ARM (10 cores)

Descriptor Type	**Match Method**	**Metric**	**MAP@1**	**MAP@5**	**Time**
ORB	BF	L1	58.33%	59.26%	∼1'
		L2	52.78%	54.03%	∼1'
		Hamming	58.33%	58.89%	∼1'
		Hamming2	58.33%	60.28%	∼1'
	FLANN	-	52.78%	54.03%	∼0.5'
SIFT	BF	L1	50.00%	52.08%	∼2'
		L2	52.78%	53.47%	∼2'
		Hamming	27.78%	28.47%	∼2'
		Hamming2	30.56%	31.94%	∼2'
	FLANN	-	52.78%	53.47%	∼0.75'
AKAZE	BF	L1	61.11%	61.81%	∼6'
		L2	61.11%	61.11%	∼6'
		Hamming	61.11%	62.04%	∼6'
		Hamming2	61.11%	61.11%	∼6'
	FLANN	-	60.04%	61.11%	∼3'

Table 6. Map@k ($k = 1$ and 5) score for several thresholds on the minimum number of matches.

Threshold	**1**	**5**	**10**	**20**	**30**
MAP@1	58.3	72.22	80.55	**80.55**	75.00
MAP@5	60.27	77.87	81.48	**81.94**	76.39

From Tables 4, 5 and 6, we concluded:

a. Compared with the rest of descriptors, text-based descriptors did not provide satisfactory results (small MAP@k) mainly because the recognized text was the name of the painter and several pictures of the same painter can be found in the DDBB. Thus, although the text embedded in the images of the query data-set was well recognized (Avg. Text Distance of 2.83), the most similar picture could not be found.
b. The texture-based descriptors perform clearly better than the text-based descriptors.
c. The proposed color-based descriptors yields a considerably better performance than the text and texture-based descriptors, mainly because it is rotation invariant.
d. Keypoint descriptors provides the best results for the query data-set.

We remark that a combination of different similarity measures into a single one can be used as an additional method for enhancing the performance of the query retrieval system. In this sense, combining several descriptors with the appropriate relevant weights can improve the effectiveness of the proposed method. However, the combination of features is a difficult task and the most appropriate weights for each descriptor should be found to obtain the right similarity between the query image and the DDBB. Under this scenario, as a future step for retrieving correctly the query image, the appropriate relevance feature weights should be calculated.

5.6 Art Gallery

In this art gallery, we present the results of our two clustering approaches, previously described in Sec. 4.4.

On the one hand, in Fig. 9 it is presented the art collection generated with texture-based features and KMeans. On the other hand, the Art collection generated with keypoint-based features and Hierarchical Clustering is shown in Fig. 10.

As can be observed in Figs. 9 and 10, 5 rooms where designed by taking into account different images parameters.

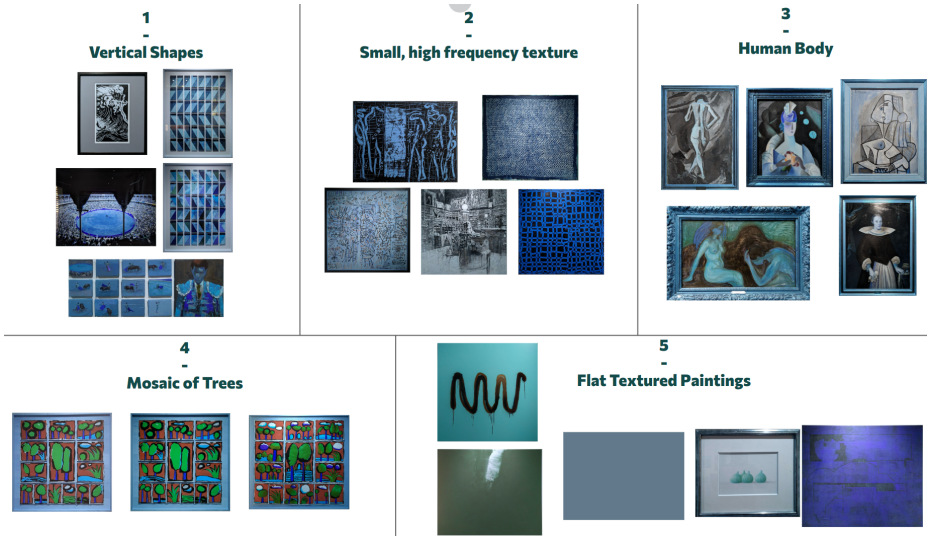

Fig. 9. Art collection generated with texture-based features and KMeans.

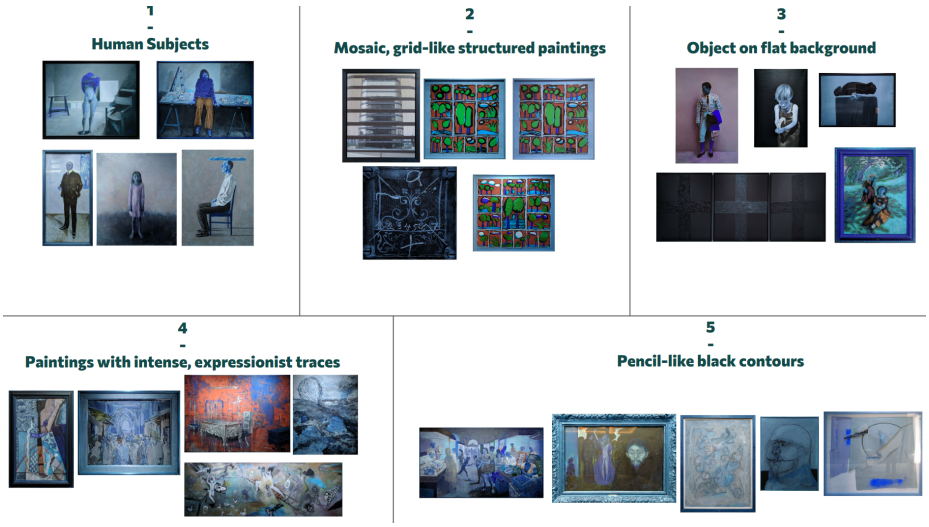

Fig. 10. Art collection generated with keypoint-based features and Hierarchical Clustering.

6 Conclusions

In this manuscript, we proposed several methods and techniques for building a simple image retrieval system for finding paintings in an image collection from three museums. We briefly described and also implemented several techniques about image preprocessing and their analysis about each process to determine the best solution application to image retrieval museum. Moreover, different types of descriptors and similarity measures were analyzed. We have seen the relationship between quantity and computational cost of the different algorithms, and their mathematical performance, trying to optimize them through the available hyperparameters. The proposed pipeline was resistant to variations in illumination, noise, color, rotations, and camera pose. This project was completed with very good results. However, the image retrieval system could be improved by enhancing text detection and mask creation, and avoiding shadow problems.

6.1 Future work

One of the possible considerations for improving the image retrieval system is a better combination of descriptors. There are different strategies applied in the state-of-the-art, but we only consider one of them. On the other hand, text detection could be improved to be invariant to the background color of the text box, using more advanced algorithms or another type of strategy as, for example, with morphological operators. The biggest improvement would be to enhance the used masks in order to crop just the paintings (i.e., the relevant information).

Note that a poorly designed mask will give a poor result, as it adds noise if the mask does not focus well on the frame or even the mask may cut relevant parts of the image (painting). For this, we could combine some of the methods described in the previous sections, such as the Canny Detector, Hough, and morphological operators.

Bibliography

[1] Ahmed, N., Natarajan, T., Rao, K.: Discrete cosine transform. IEEE Transactions on Computers **C-23**(1), 90–93 (1974). https://doi.org/10.1109/T-C.1974.223784

[2] Ali, J.: A comparative study of various types of image noise and efficient noise removal techniques (2013)

[3] Calonder, M., Lepetit, V., Strecha, C., Fua, P.: Brief: Binary robust independent elementary features. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) Computer Vision - ECCV 2010. pp. 778–792. Springer Berlin Heidelber, Berlin, Heidelberg (2010)

[4] CANNY, J.: A computational approach to edge detection. In: Fischler, M.A., Firschein, O. (eds.) Readings in Computer Vision, pp. 184–203. Morgan Kaufmann, San Francisco (CA) (1987). https://doi.org/https://doi.org/10.1016/B978-0-08-051581-6.50024-6, bluehttps://www.sciencedirect.com/science/article/pii/B9780080515816500246

[5] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). vol. 1, pp. 886–893 vol. 1 (2005). https://doi.org/10.1109/CVPR.2005.177

[6] Immerkær, J.: Fast noise variance estimation. Computer Vision and Image Understanding **64**(2), 300–302 (1996). https://doi.org/https://doi.org/10.1006/cviu.1996.0060, bluehttps://www.sciencedirect.com/science/article/pii/S1077314296900600

[7] Lowe, D.G.: Object recognition from local scale-invariant features. Proceedings of the Seventh IEEE International Conference on Computer Vision **2**, 1150–1157 vol.2 (1999)

[8] Otsu, N.: A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics **9**(1), 62–66 (1979). https://doi.org/10.1109/TSMC.1979.4310076

[9] Pavlidis, T.: Algorithms for graphics and image processing. In: Springer Berlin Heidelberg (1982)

[10] Pietikäinen, M., Zhao, G.: Two decades of local binary patterns: A survey. In: Advances in independent component analysis and learning machines, pp. 175–210. Elsevier (2015)

[11] Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: Proceedings of the 9th European Conference on Computer Vision. pp. 430—443 (2006)

[12] Rublee, E., Rabaud, V., Konolige, K., Bradski, G.R.: Orb: An efficient alternative to sift or surf. 2011 International Conference on Computer Vision pp. 2564–2571 (2011)