Report

EEE3032
Computer Vision and Pattern Recognition

Visual Search of an Image Collection

Student :
Antoine Edy *(6797948)*

Professor :
Miroslaw Bober

November 14, 2023

Abstract

The digital landscape is currently witnessing a significant transformation in the way users interact with visual content. In recent years, a notable surge in demand has emerged for the ability to search for images based on their visual appearance [3]. This report delves into some of the ways to accomplish visual search using computer vision methods.

This coursework aims at developing programs capable of conducting visual searches within an image collection. With an image as a search query, the programs should provide a ranked list of images from the collection based on their similarity to the query image.

In order to compare the multiple images, we need to retrieve some information that they contain known as the features of the images, such as the colours of the image (*Global Colour histogram*) or the main orientations of the edges (*Edge Orientation histogram*). We can also increase the number and precision of the information retrieved by creating a spatial grid and comparing the features of the images in each tile of the grid, therefore adding a geographical comparison between the images of the dataset (*Spatial Grid, with colour and/or texture*). These methods of feature retrieval have some parameters, which values can be chosen : we will study their impact on the result and try to find the optimal ones.

When computing a distance between two images (so two descriptors), multiple metrics can be chosen. We will go through some of them and study their impact on performance.

We will also explore a dimensionality reduction method, the PCA (Principal Component Analysis), and a distance metric which share a lot of ideas with it, the Mahalanobis distance, in order to see if the performance improves or if we can retrieve interesting things from the plotting that we will be able to do in a low dimension ($dim \leq 3$).

Furthermore, we will implement two other techniques:

- the Bag of Visual Words retrieval, another image retrieval technique,
- the SVM or Support Vector Machine, to classify the images. We will implement a binary classification first (between two sets of images), and then a multiclass classification.

Lastly, in order to compare the different methods used, we will need to implement an evaluation methodology, using different metrics, such as the precision and recall or a confusion matrix.

The MATLAB code associated with this report can be found in the folder name `code` attached. The files are explicitly named and commented to make the code understandable for the reader.

Contents

1 Description of visual search techniques implemented

1.1 Global Colour histogram

A global colour histogram is a representation of an image's colour distribution. It quantifies the frequency of different colour values in an image. This technique is primarily used to capture the overall colour characteristics of an image.

To create a colour histogram, an image is often divided into a set of discrete colour bins. Each pixel in the image is then assigned to one of these bins based on its colour value. We will be using the RGB (*Red Green Blue*) colour space here, but others such as HSV (*Hue, Saturation, Value*) exist. We need to choose a quantization: if we call q the number quantization, the values of r, g, b are now in $\{0, 1, ..., q\}$. Then, q^3 different combinations can be made from these new values. We create the bins as following : $value = r * q^2 + g * q^1 + b * q^0$. So $value \in [0, q^3 - 1]$.

After computing the value for all the pixels, a histogram is generated by counting the number of pixels in each bin. The result is a histogram that represents the distribution of colours in the image. Therefore, the distance between two images is the difference of the two histograms associated.

1.2 Global Edge Orientation histogram

The global edge orientation histogram is a technique used to capture the overall edge structure or orientation information within an image. It is useful for identifying images based on their textural and structural characteristics.

To create an edge orientation histogram, you need to perform edge detection on the image. In our case, we will use the Sobel operator [2], which highlights the edges in an image by detecting rapid changes in pixel intensity. We can compute the edge orientation with the Sobel orientation the following way:

- Compute the gradient in every pixel using a convolution matrix in both directions x and y:

$$\frac{\delta f}{\delta x} = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, \frac{\delta f}{\delta y} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

- Compute the edge direction:

$$\theta = \arctan(\frac{\delta f}{\delta y} / \frac{\delta f}{\delta x})$$

After detecting edges, the orientation of edges is quantified. This information is then used to construct a histogram, where the bins represent different edge orientations. We also have to decide how many divisions we want. For example, if we want 4 zones, because $\theta \in [-\pi/2, \pi/2]$, the dividers will be $\{-\pi/2, -\pi/4, 0, \pi/4, \pi/2\}$, creating 4 different segments.

1.3 Spatial Grid, with colour and texture

For the spatial grid method, we divide each image into smaller regions, and each region is considered as an image in which we retrieve the features explained in the previous section. We then compare the descriptors of every region between two images. The advantage of this method is that it adds a geographical aspect: the location of colours and orientations matters this time.

We can choose what features to compute in each region : colours, edges... or even both. We can even add coefficients to give more importance to colour or region. These coefficients can be changed depending on the image we are trying to retrieve. This will be tackled in section 2.

1.4 The use of Principal Component Analysis (and Mahalanobis distance)

Principal Component Analysis (PCA) [6] is a dimensionality reduction and data transformation technique. It is used here to reduce the number of features while preserving most of the information. PCA accomplishes this by transforming the original dataset of features into a new coordinate system, so new features, in which the variance is maximized along the principal components. The new features are less numerous than the previous ones and are a combination of the previous' most important one (important here mean that it discriminates well the images).

The Mahalanobis distance is closely related to the PCA [1].The distance between $\vec{x}$ and $\vec{y}$ can be computed like so:

$$d_M(\vec{x},\vec{y};Q) = \sqrt{(\vec{x}-\vec{y})^\top S^{-1}(\vec{x}-\vec{y})} \quad \text{with} \begin{cases} Q \text{ the set of elements} \\ S \text{ the covariance matrix associated} \end{cases}$$

The Mahalanobis distance can handle datasets with non-spherical clusters, where the spread of data points vary along different directions. In other words, it captures the anisotropic nature of the data, which is not the case of the other distance metrics.

1.5 Bag of Visual Words

The Bag of Visual Words (BoVW) [5] is a popular technique used in computer vision and image analysis for representing and analysing the content of images. To compute BoVW, we follow these steps:

- Feature Extraction: we will use the SIFT (Scale-Invariant Feature Transform) keypoints to extract the features. They capture distinctive characteristics in different regions of the image.

- Codebook Building: the codebook is created by clustering the extracted features from a set of training images. We will use k-means to group similar features into clusters (visual words).

- Feature Quantization and Histogram Representation: each feature extracted from an image is assigned to the nearest visual word in the vocabulary (cluster). As a result, each image is represented as a histogram, with each bin corresponding to a

visual word and the bin's value indicating how many times that visual word is found in the image.

The main advantage of BoVW is that it is robust to changes in scale, orientation, and lighting conditions, making it suitable for object recognition and image retrieval.

1.6 Classification using Support Vector Machine

Classification using Support Vector Machines (SVM) is a popular machine learning technique for binary and multiclass classification tasks. SVMs will be used to find decision boundaries that maximize the margin between different classes in a dataset. The SVM algorithm aims to find the optimal hyperplane (or decision boundary) that maximizes the margin between the two classes and minimize the classification error.

Support vectors are the data points that are closest to the decision boundary (the margin): they play a critical role in defining the margin and the separating hyperplane.

2 Experimental results

For the following part, the results will be displayed the following way.

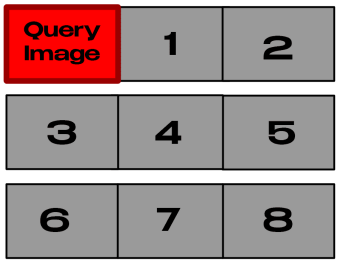


Figure 1: Illustration on how to read the results

The query image will be in the top left corner, the closest one to the query image on its right (number 1 on the image), and so on. The furthest in the list will then be in the bottom right corner (number 8 on the figure).

2.1 Global Colour histogram

2.1.1 The general results

For now, we will choose the number of quantization $q = 4$, which means $4^3 = 64$ bins, but we will discuss this number later on (see section 2.1.3). To have an overview of the results of the Global Colour histogram method for the entire dataset, we computed the evaluation (precision and recall) for 6 images of every class, so a total of $6 * 20 = 120$ images. The results are the following:

	Precision (in %)	Recall (in %)
Mean	15.3	7.5
Best	67.1	34.8
Worst	0	0

Table 1: Precision and recall of the global colour histogram

The Global Colour histogram method is one of the simplest descriptor to compute. The results we obtained here are disappointing : the precision is low. For a random image, this method gives bad results, especially when the colour of an image does not help identify what it represents. However, this method suits some images pretty well, as we can see with an image which scored 67.1% of image retrieval (see section 2.1.2).

We can then plot the confusion matrix:

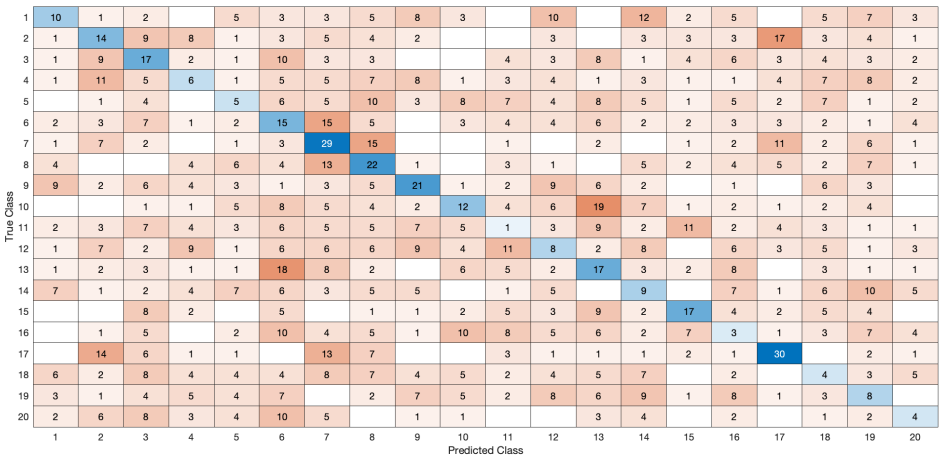


Figure 2: Confusion matrix for the global colour histogram method

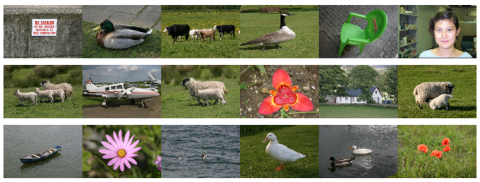
We can see where the program fails the most. For example, some images from class 7 are confused with class 8, same for 2 & 17 or even 10 & 13 (the dark red tiles from the confusion matrix).

When we look at the dataset, we see that class 7 contains cars and 8 bicycles. We can notice that both of them are often in an urban environment, with a lot of grey ! Both the class 2 (trees) and 17 (roads) contain a lot of nature, so green colour, and both 10 (flowers) and 13 (books) are very colourful, so the colour bins might be balanced.

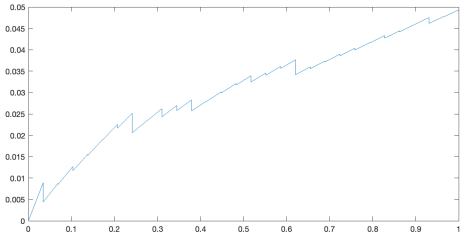
2.1.2 Two examples to illustrate the method

As seen with the confusion matrix on figure 2, for some classes (3, 13, 17), most predictions are correct. For others, (5, 11, 20), the method is not working at all. Here are two examples to illustrate this result.

First, an example of an image representing a street sign for which this method is inefficient.



(a) The 18 closest images



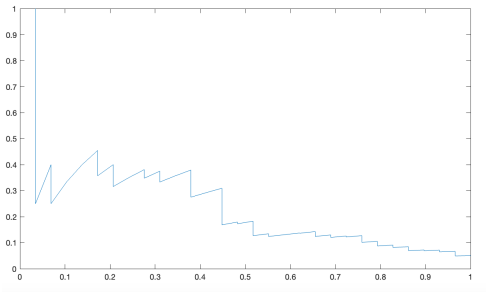
(b) P/R curve

Figure 3: An example of bad result with the global colour histogram

In this image, the prominent grey colour of the background distort the results and leads to finding images with grey ducks and grey water, and the red letters lead to finding red flowers. We compute a very low score of $precision = 0\%$ and $recall = 0\%$.

However, this method applies better on specific types of images, like the following.

(a) The 18 closest images



(b) P/R curve

Figure 4: An example of a good result with the global colour histogram

Images containing trees often contain a lot of green for the leaves, some brown for the log and a lot of blue for the sky. Therefore, this visual search method has some good results. The evaluation is better, with $precision = 35.3\%$ and $recall = 20.7\%$, but even for the images that are not trees, the images found are very similar (figure 4a).

2.1.3 The effect of q, number of quantization

q is the number of quantization, which is a hyperparameter that the user can choose. We can compute the evaluation with the same method as before, and with different values of q to compare the performance.

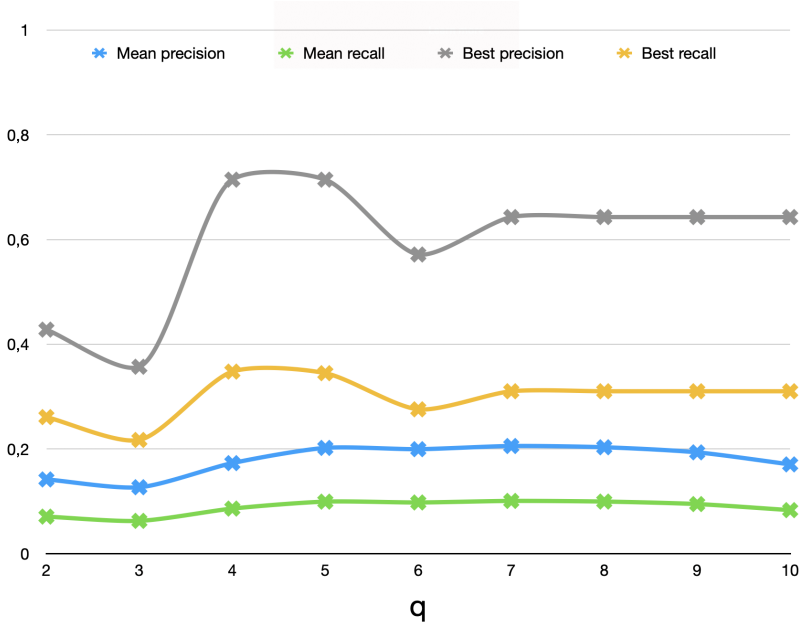


Figure 5: Precision and recall for $q \in [2, 10]$

The optimal choice for q seems to be 4 or 5. How can we explain this result ? If we use a low value for q, then two colours which are different might end up in the same bin. For example, with $q = 3$, the colours $[100, 100, 100] \leftrightarrow [1, 1, 1]$ and $[150, 90, 170] \leftrightarrow [1, 1, 1]$. So these two will be in the same bin while they are very different (see figure 6). The same goes for q too high. With $q = 9$, $[40, 216, 109] \leftrightarrow [1, 6, 3]$ and $[26, 216, 109] \leftrightarrow [0, 6, 3]$, so different bins while these colours are almost the same.

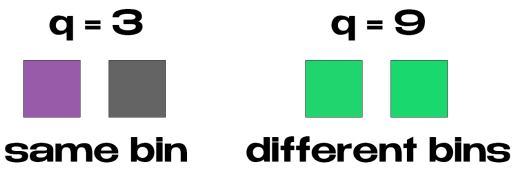


Figure 6: Illustration of the reason why q matters

The same problem occurs for edge orientations.

2.2 Spatial Grid (Colour and Texture)

2.2.1 The hyperparameters

As explained in a previous section (see 1.3), the spatial grid method is computing a colour histogram and an edge orientation histogram for regions of the image. This method requires at least 3 hyperparameters to be chosen : the number of bins of the colour histograms, the number of bins of the edge orientation histograms and the size of the grid (number of zones of the image). We've seen in the previous section (see 2.1) that the optimal number of bins is 4^3 for the colours. We will take the same number of bins for the edge orientation to facilitate the computations. For the grid, we will start with a

$5 * 5$ grid, so 25 zones of the image, and change this number in section 2.2.4 to see how the grid size impact the results.

To these hyperparameters, we can add another one to give a certain weight either to the colours or to the edge orientations. We will call this parameter α (as it is in the code). The distance between two images x and y will be computed this way:

$$d_{(x,y)} = \alpha * d\_colours_{(x,y)} + (1 - \alpha) * d\_eo_{(x,y)}$$

This means that $a = 1$ means that the program is equivalent to a grid colour histogram, and $a = 0$ to a grid edge orientation histogram.

2.2.2 The general result

	Precision (in %)			Recall (in %)		
α	0	0.5	1	0	0.5	1
Mean	25.5	15.1	11.1	12.4	7.3	5.3
Best	100	71.4	57.2	48.3	34.5	27.6

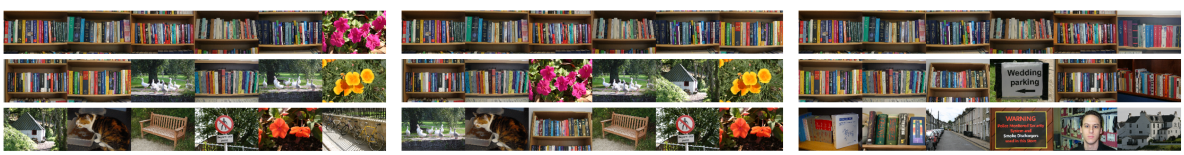
Table 2: Precision and recall of the grid histogram method for different values of α

We can see an increase of performance compared to the results of the global colour histogram method, as expected. Moreover, it seems that the edge orientations are more discriminative, meaning that they can differentiate the classes more and therefore give better results. These general results also show that the performance is heterogenous between the images; let's delve into three examples to understand why it is so.

2.2.3 Three examples to illustrate

Let's analyse two examples to understand better what really happens behind the computations. The first example is that of a library. What differentiate the photos of libraries and the others are :

- a little bit the diversity of colours (but we can see that colourful flowers are mistaken on figure 7a)
- mostly because they have the same edge orientation : the main lines of the image are vertical.



(a) $\alpha = 1$ colour only

(b) $\alpha = 0.5$ both

(c) $\alpha = 0$ edge orientation only

Figure 7: Grid histogram for 3 values of α

We can conclude that for this image, a grid of edge orientation histogram is very efficient. The same happens with the planes:

(a) $\alpha = 1$
colour only

(b) $\alpha = 0$
edge orientation only

Figure 8: Grid histogram for 3 values of α

Edges give an almost perfect precision of 94.2% precision, while colour gives a 0% precision.

Doing the same computations for cats, we find the following results:

	Precision (in %)			Recall (in %)		
α	0	0.5	1	0	0.5	1
Books	70.6	47.1	41.2	41.4	27.6	24.1
Planes	94.2	0	0	55.2	0	0
Cats	0	5.9	0	0	4.3	0

Table 3: Precision and recall of the grid histogram method for different values of α and different images

This table shows the diversity of the results on the images : on two of them, planes and books, the visual search using grid histograms gives satisfying results. On the cats, however, because their shape and colour might diverge a lot, the results are disappointing. We will now see more advanced visual search techniques to, hopefully, have better evaluation on average.

2.2.4 Finding the best grid size

We will choose the following parameters : $\alpha = 0.5$ and the number of bins for both histograms is $n_{bins} = 4^3 = 64$.

Grid size	Mean precision (in %)	Mean recall (in %)	Time of computation
$2^2 = 4$	14.1	6.8	20s
$3^2 = 9$	15.6	7.5	1min
$4^2 = 16$	14.7	7.1	2min57s
$5^2 = 25$	15.1	7.3	3min40s
$6^2 = 36$	14.2	6.8	5min55

Table 4: Precision and recall of the grid histogram method for different grid sizes

I added the time of computation of the descriptors with my computer as well (MacBook Pro M1), because it has a significant impact on the optimal value of the grid size. It seems that it is not worth to increase the grid size too much: a 3 by 3 grid is optimal in our case. For a small number of zones (like a 1x1 or 2x2 grid), the result lacks of geographical understanding of the images. But for a large number of zones (6x6 for example), the grid might compare zones of the image that are too small to be relevant, as explained in the following illustration.

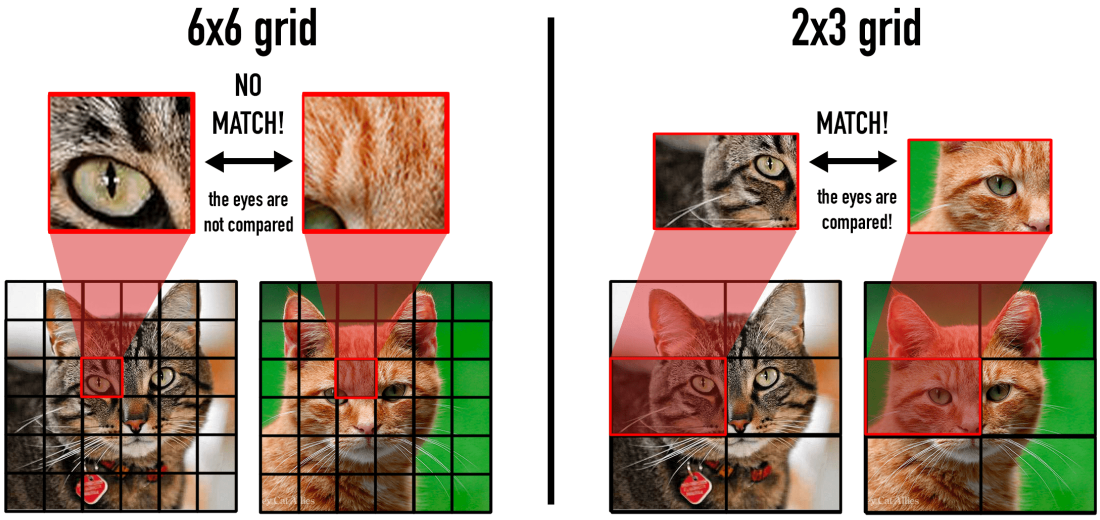


Figure 9: Illustration of the reason why a bigger grid can lead to bad results

2.3 Using PCA

2.3.1 With the Euclidean distance

We will now use PCA to project the image descriptors into a lower dimensional space. We will apply PCA to the descriptors computed with a grid edge and colour histograms. The number of bins for each histogram is $4^3 = 64$, so $2*64 = 128$ for each cell (colours and edge orientation). The size of the grid is $4^2 = 16$. Overall, each image has $16*128 = 2048$ descriptors. PCA will help reduce this number, by combining these descriptors to keep n descriptor (n to be chosen).

Number of desc.	Mean precision	Mean recall	Best Precision	Best recall
No PCA	14,71	7,1	78,6	37,9
10	13,1	6,3	42,9	20,7
30	12,8	6,1	50	24,1
50	12,8	6,1	50	24,1
70	11,3	5,4	57,1	27,6
150	11,7	5,6	64,3	31

Table 5: Precision and recall for different dimensionality reductions using PCA

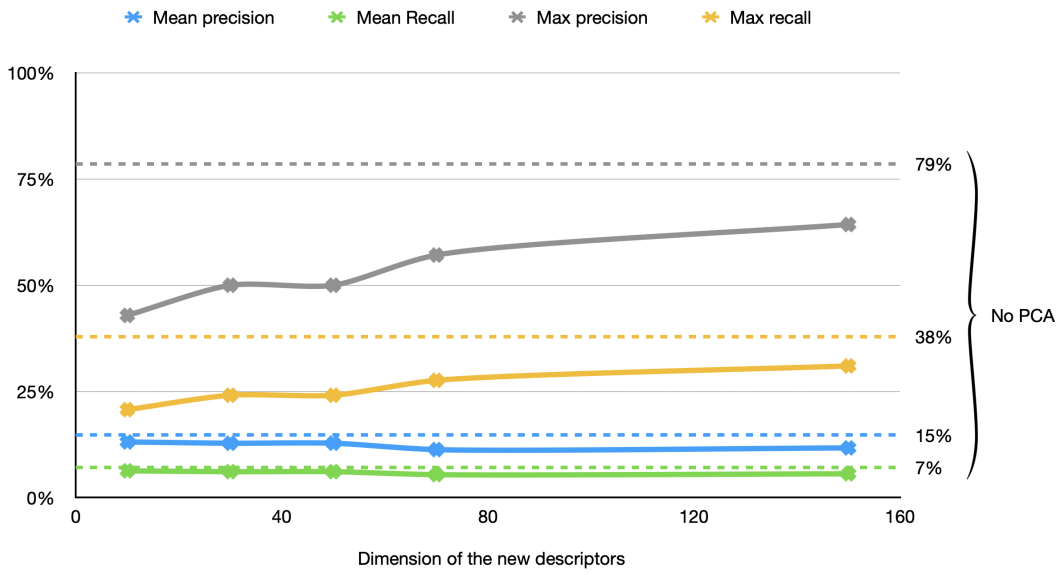
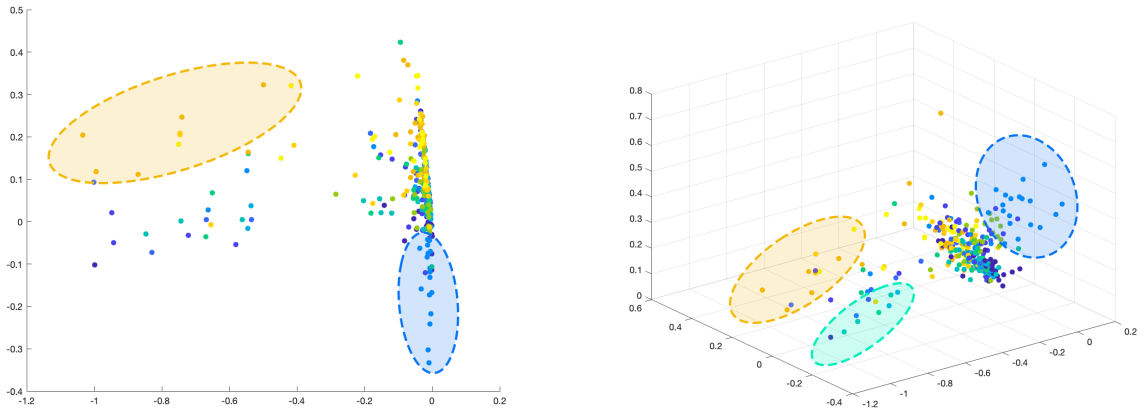


Figure 10: Plot of precision and recall for different dimensionality reductions using PCA

We can notice a slight decrease in performances with PCA compared to without. In fact, PCA tries to keep most of the information condensed into a few parameters : it is logical that the precision decreases. The result with PCA is still interesting. While the dimension goes from 2048 to 150 (so a decrease of 87%), the best precision goes from 78.6% to 64.3% (a decrease of 22%). For a very large number of images, having such a decrease in features while keeping a good precision is useful for computation time and power.

2.3.2 PCA visualization

We can reduce the dimensions of the descriptors to $d = 2$ or $d = 3$ to plot the images and analyse the plots.

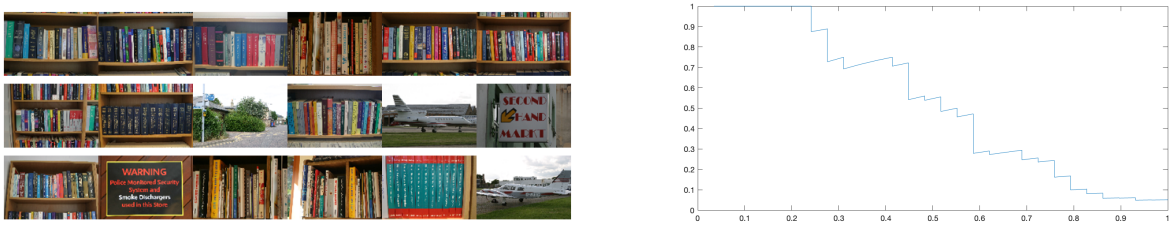


(a) Plot in 2 dimensions

(b) Plot in 3 dimensions

Figure 11: Images plotted in 2D and 3D using PCA

Here, each point corresponds to an image, with the dimensions reduced to 2 and 3 respectively, the colour corresponding to the classes. We see on figure 11 that both in 2D and 3D, some clusters can be noticed, meaning that enough information is stacked into 3 descriptors only to differentiate some of the classes. The blue one on the figure corresponds to class 13, which is a class of books. Let's see if the evaluation of one of the image of this class is coherent with these plottings.



(a) The results with 3 features only

(b) The P/R curve associated

Figure 12: Good results with PCA ($d = 3$) for the books

With a precision of 70.6% and a recall of 41.4%, the books are very well distinguished from the other images, and all this with 3 descriptors only. This might be explained by the fact that little information is need to recognize books : only vertical lines. This presumption is validated with images of signs (with dominant vertical lines) being close to books as well.

2.4 Using different distance metrics

We will implement different distance metrics and see their impact on the precision and recall. We will use the descriptors from the grid edge orientation histograms method, which have shown to have the best results so far. Here are the results after computation:

Distance metric	Mean precision	Mean recall	Best Precision	Best recall
l_1 (Manhattan)	28.0	13.6	100	48.3
l_2	24.1	11.7	100	48.3
l_3	21.7	10.5	100	48.3
l_4	20.5	9.9	85.7	41.3
l_{inf} (Chebyshev)	18.5	9.0	100	48.3
Mahalanobis	22.2	10.7	92.9	44.8
Canberra	27.8	13.5	100	48.3

Table 6: Precision and recall for different measure metrics (grid of edge orientation histograms)

I added the Canberra distance, which can be computed this way :

$$d_{(x,y)} = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|}$$

It is interesting to notice that l_1 distance tends to perform a little bit better than l_n with $n \geq 2$. l_1 distance, being the sum of absolute differences, is less sensitive to outliers than l_2 distance, which involves squaring the differences. The data having outliers, l_1 is more robust and performs slightly better.

2.5 Bag of Visual Words retrieval

We implement the Bag of Visual Words using MATLAB functions, which are part of the Computer Vision Toolbox [4].

2.5.1 General results

We implemented a BoVW system using the SIFT keypoint detector, the SIFT descriptor and the k-Means algorithm to create the codebook. The grid histogram used for comparison is the one combining colours and edge orientations, with the same weight (i.e. $\alpha = 0.5$, α define in the section 2.2.1).

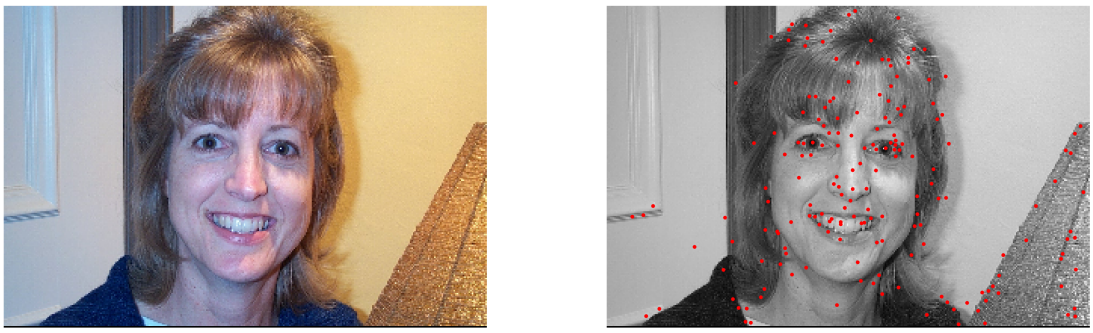
Method	Mean prec.	Mean recall	Best prec.	Best recall
Grid Histogram	14.7	7.1	78.6	37.9
Bag of Visual Words	21.9	10.5	100	48.3

Table 7: Result of the BoVW compared to the grid histogram

We notice a performance increase of 32% on the average precision using this technique.

2.5.2 Two examples in details

The first step of this method is to find the key points of the images. Our example will be a human face.



(a) The image selected (b) The SIFT keypoints computed

Figure 13: An example of SIFT keypoint on a face

From these keypoints, features are extracted and bags of words created. When comparing the bags of words, we obtain these results:

(a) The results (b) The P/R curve

Figure 14: Result of the BoVW image retrieval on a human face

We obtained a precision of 23.5% and a recall of 13.8%. All faces contain different shapes and colours, and the main lines of a human face might vary a lot depending on the lights, the quality of the photo and the surroundings, which can explain the low result. Let's try with an object that we can easily identify visually, no matter what: bicycles.

(a) The image selected (b) The SIFT keypoints computed

Figure 15: An example of SIFT keypoint on bikes

(a) The results (b) The P/R curve

Figure 16: Result of the BoVW image retrieval on bikes

The evaluation goes up to 94.1% of precision and 55.2% of recall. These astonishing results are explained by the fact that only few keypoints are necessary to perfectly recognize a bicycle: some on the two wheels and on the handlebar might be enough to perfectly describe it !

2.6 Object classification using SVM

SVM is not really proper visual search but more classification. We will use the SIFT descriptors that we computed before.

2.6.1 Binary classification

We first try binary classification using SVM. This program consists in training an SVM to differentiate two classes. The example chosen is that of cows (class 5) and cars (class 7). Each of these classes have 30 images. We will create a train database with 15 elements of each class, so 30 images, and a test database, with the 30 others.

We train the SVM with the train database, and compute the precision with the test one. We get a precision of 93.3%, which means that for most images, the model rightly found its class.

2.6.2 Visualizing the boundaries

An interesting thing to plot while using SVM is the decision boundary. In order tu visualize a linear decision boundary, we need to plot our data in 2D. To do so, we will use the PCA algorithm to reduce the number of dimensions to 2. Then, we will train another SVM model, and plot the data and the decision boundary. We can find the decision boundary equation like so:

$$y = -(\frac{\alpha}{\beta} * x) - \frac{B}{\beta} \quad \text{with} \begin{cases} \{\alpha, \beta\} \text{ the linear predictor coefficients} \\ B \text{ the bias term} \end{cases}$$

Let's plot the result, where colours represent classes:

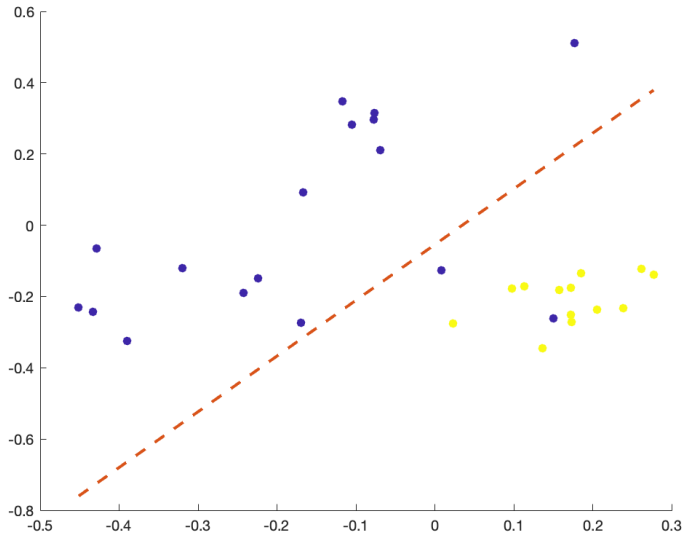


Figure 17: Plot of images in 2D (with PCA) and the decision boundary made with SVM

Even with only two features, we still find a very good precision of 93.3%! If this works so well, it is because we applied PCA only on the two sets of images. But this is a way to visualize the boundary decision.

2.6.3 Multiclass classification

For multiclass classification with SVM, we will use the same technique of creating a train dataset with 20 images per class, and a test dataset with the rest (so approximately 10 images per class). We compute a 37% precision. Let's plot the confusion matrix:

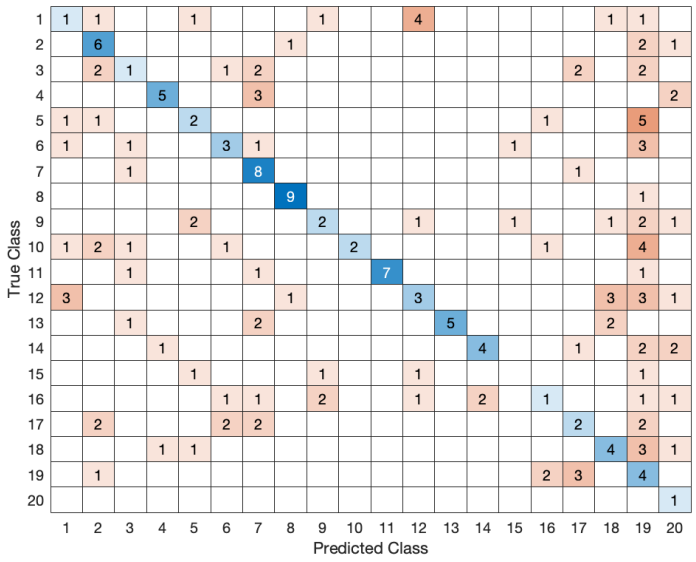


Figure 18: Confusion matrix for the multiclass SVM

The classes in the dataset have high overlap or are not easily separable, so SVM struggles to achieve a high precision.

3 Conclusions drawn from experiments

3.1 Overview of the experimentations

Here is a summary of the results we obtained using the different methods:

Method used	Mean precision	Best precision
Global colour histogram	15.3	67.1
Grid colour histogram ($\alpha = 1$)	11.1	57.2
Grid c. & e.o.* histogram ($\alpha = 0.5$)	15.1	71.4
Grid edge orientation histogram ($\alpha = 0$)	25.5	100.0
Grid c. & e.o. with PCA (10 features)	13.1	42.9
Grid c. & e.o. with PCA (150 features)	11.7	64.3
Grid e.o. with l_1 distance metric	28.0	100.0
Bag of Visual Words retrieval	21.9	100.0
Binary SVM classification	90.0	100
Multiclass SVM classification	37.0	93.3

Table 8: Summary of the methods used

**Grid c. & e.o: Grid colour and edge orientation.*

Overall, the best methods for image retrieval for our dataset are the grid edge orientation histograms, Bag of Words and SVM classification. These methods are not necessarily the best ones for every dataset, but the results show that it suits ours quite well.

3.2 Choosing the right method for a given image

The one thing we can notice from the results of table 8 and from the whole report is that the results, for every method, are highly dependable on the image of the dataset. Almost every method used is well suited for certain images, and not at all for others. In other words, the results are highly heterogenous. In general :

- Global colour histogram works well with images that have few colours and no specific patterns (trees, grass),
- Grid edge orientation histogram is for images with specific shapes that perfectly describe what is represented (planes, books),
- Bag of Visual Words retrieval is efficient when the variations in visual content are relatively constrained or consistent. This is the case for bicycles.
- SVM gives astonishing results in binary image classification tasks where the objective is to distinguish between two classes only.

The database used for this coursework can be considered small (in comparison to most of the real-life uses of image retrieval). The results of this report illustrate the principle that SVM works well in small-scale databases (it can efficiently handle the classification task without overfitting), while Bag of Visual Words retrieval works best in large-scale databases (limited vocabulary of visual words and overfitting).

4 Bonus work: find image by drawing

4.1 The aim of the programme

From the beginning of this coursework, the aim was to find the images that were the closest to a query image. But what if you don't have a query image ? What if you just want, or example, to find an image where the cow is in the top left corner ? Or positioned a certain way ? Lying down ? Or taking the entire image ?

The aim of my programme is to take as a query image not an actual image of what you want, but an image describing where you want a specific object (we will choose cows in this section) to be on the image. For example, if we want a cow lying down in the centre of the image, we can give the programme the following image query:

(a) The query image (b) The expected result

Figure 19: What the programme is intended to do

The user draws in white where he wants the cow to be on the image. This becomes the query image. On the left, the programme gives the image that best corresponds to the query, and can also provide a set of the closest images.

4.2 The different steps of the algorithm

The idea behind this programme is to be able to separate the cow from the rest of the image (the grass, the sky, some water maybe), and then compare the query image with the cow to see if it matches the position. Here are the main steps detailed:

1. Divide the pixels of and image of the dataset into different groups to separate the main zones of the images. We will use the RGB values of each pixel, and divide them into 3 clusters using a k-means algorithm.

2. For each group found (3 in our case), create the image where the group is in white and the rest in black.

3. Compute a distance between the query image and each of the images generated during the last step. The smallest distance is kept, and will be the distance between this image of the dataset and the query image.

4. Reiterate for every image of the dataset and rank the images on their distance to the query image.

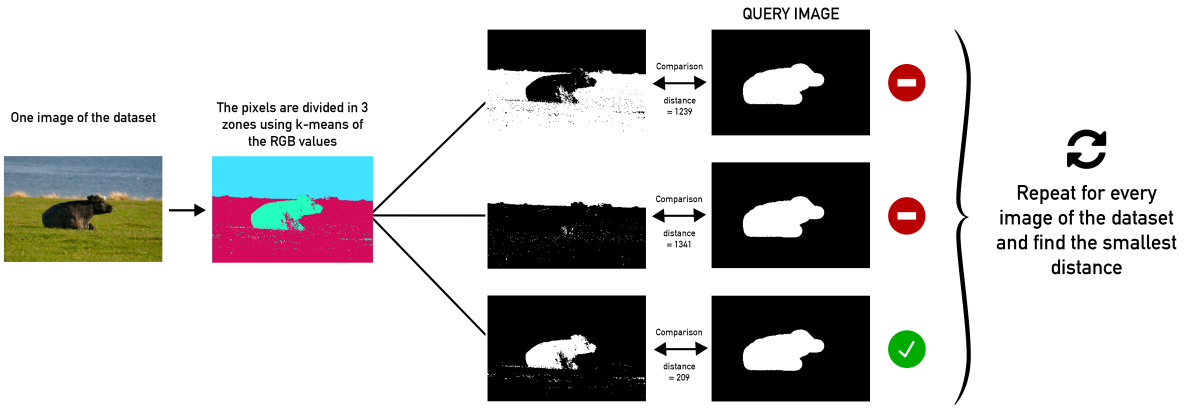


Figure 20: The steps of the method

4.3 The overall results

The dataset contains a set of images that had been unused for now, containing the position of every different thing that is on the images (named `GroundTruth`). To compute a precision evaluation, we can select the cow on every of these images and use it as a query image. Then, we can see if the algorithm manages to find the image that has been used to generate that GroundTruth.

For the example of the cow images, we find a result of 56.7%, which means that in 56.7% of the time, the first image found is the one that was used to create the GroundTruth image.

I also tried with the sheeps, for a result of 50% precision.

4.4 A few illustrations

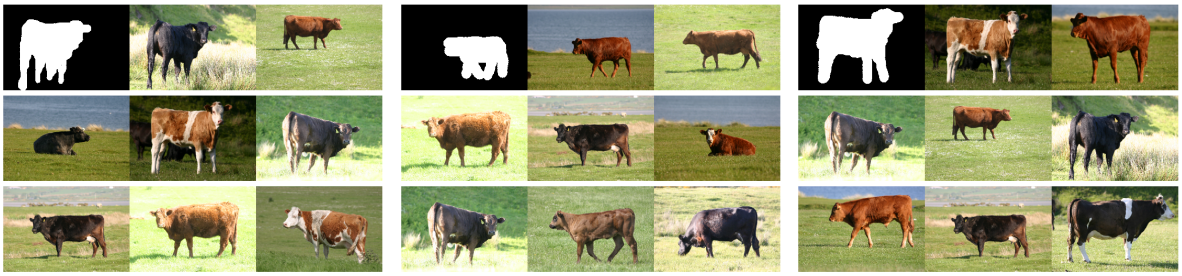


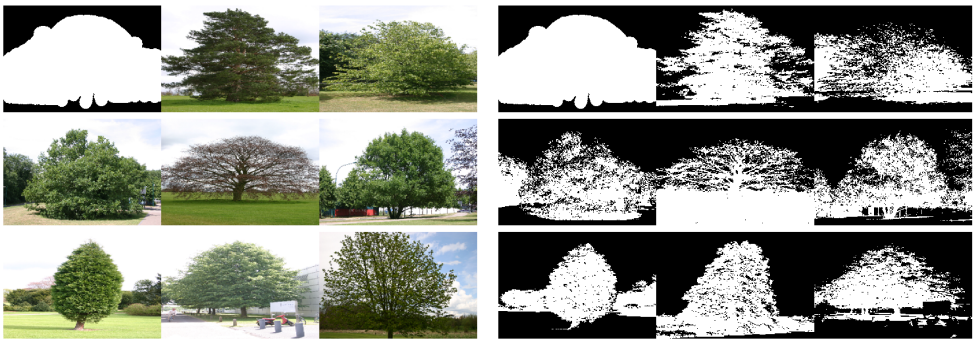
Figure 21: 3 successful results

We can see that not only did the algorithm found the right image associated with where the cow is, but the other images close to them share similar points. For the first and third example, they almost all face the right direction, while it is the opposite for the second example.

However, this technique has some issues that we have to analyze.

4.5 Limitations of the algorithm

In order for this algorithm to have good results, the object needs to be perfectly identifiable thanks to its colour. If not, the k-means technique applied on the RGB values of the pixels will not identify the object clearly. The example of the trees is one of them: because the trees have the same colour as the grass or some other elements of the nature in the background, the algorithm has a hard time delimitating it efficiently. Therefore, the precision decreases to 20% for the trees.



(a) Result for one query

(b) Visualization of the tree retrieval for each image

4.5.1 How can this algorithm be useful

This kind of algorithm can be improved by using a more complex algorithm to retrieve the object you look for.

I can clearly see a use case in for search engines: it can enhance image searches by allowing users to draw what they're looking for rather than relying solely on keywords. This can be especially useful when the user struggles to describe an image in words. If you want an image of a dog in the top left corner on Google image, for example, the result is not very satisfactory. By combining the two types of search, keywords and drawings, we might find way better results.

5 Bibliography

References

[1] Richard G. Brereton. "The Mahalanobis distance and its relationship to principal component scores". In: *The chemometrics column* (2015).

[2] R. Gonzalez and R. Woods. *Digital Image Processing*. Addison Wesley, 1992.

[3] Sadiq H. Abdulhussain Ibtihaal M. Hameed and Basheera M. Mahmmod. "Content-based image retrieval: A review of recent trends". In: *Cogent Engineering* 8.1 (2021). Ed. by D T Pham, p. 1927469. DOI: 10.1080/23311916.2021.1927469. eprint: https://doi.org/10.1080/23311916.2021.1927469. URL: https://doi.org/10.1080/23311916.2021.1927469.

[4] MatLab. *Computer Vision Toolbox*. URL: https://uk.mathworks.com/help/vision/ (visited on 11/09/2023).

[5] Ravi Shekhar and C.V. Jawahar. "Word Image Retrieval Using Bag of Visual Words". In: (2012), pp. 297–301. DOI: 10.1109/DAS.2012.96.

[6] Jolliffe Ian T. and Cadima Jorge. "Principal component analysis: a review and recent developments". In: *Philosophical Transactions of the Royal Society* (2016). URL: http://doi.org/10.1098/rsta.2015.0202.