MULTIMEDIA RETRIEVAL & CLOUD COMPUTING

2022/2023

Summary

This paper will highlight the knowledge acquired during the development and hosting of an indexing and searching multimedia application on a Cloud resource.

José António Dantas Macedo
Margarida Ribeiro Cosme

Index

Introduction

In this report, we present the development and hosting of a multimedia retrieval application on a cloud resource. The application is specifically designed to index and search for images using various descriptors such as BRG, GLCM, HOG, HSV, SIFT, LBP, ORG, and some deep learning models. The goal of the project was to create a system that could efficiently search and retrieve images from a database, using advanced techniques to extract meaningful features from the media.

To achieve this, we developed an application using the Flask framework. This allowed us to take advantage of the scalability and flexibility of cloud computing, while also providing a user-friendly interface for accessing the system. Additionally, the use of several descriptor models helped us extract a more robust set of features from the images, making the retrieval more accurate and reliable.

We will discuss in detail the process of developing the application and the challenges faced during the project. Furthermore, we will evaluate the performance of our system and compare it to other similar systems. Finally, we will discuss the potential impact of this work on the fields of image retrieval and cloud computing and suggest possible directions for future research.

Part 01: Indexing application and multimedia search

The goal of this part was to develop a search engine that used a variety of descriptions

Indexing the database with descriptors

Indexing a database with descriptors is an essential step in the process of multimedia retrieval. Descriptors are mathematical representations of the features of a multimedia file, such as the color, texture, and shape of an image. By extracting these features and creating descriptors, we can then use them to search and retrieve multimedia files from the database.

The process of indexing a database with descriptors begins with the extraction of features from multimedia files. This can be done using various techniques, such as color histograms, texture analysis, and feature detection. Once the features have been extracted, they are then transformed into descriptors using mathematical algorithms. These descriptors are then used to create an index of the multimedia files, which can be searched and queried later on.

There are different types of descriptors that can be used, for example, for images, we use GLCM for texture analysis, HOG for object detection, HSV for color analysis, SIFT for feature detection, and some deep learning models for more robust feature extraction. Each descriptor has its advantages and disadvantages, and the choice of descriptor will depend on the specific requirements.

The indexing process was done in a centralized manner because of the size of the database and the computational resources available. The descriptor extraction and indexing were done on a single local machine.

BRG, GLCM, HOG, HSV, SIFT, LBP, and ORB are the descriptors that we used for image indexing.

Each descriptor has its own unique characteristics and is best suited for certain types of image analysis; we will describe their different characteristics very shortly.

BRG (Blue, Red, Green)

Is a color-based descriptor that separates an image into its red, green, and blue channels, and then it creates a histogram for each channel.

This descriptor is useful for color-based image retrieval and segmentation, allowing for the distinction of images based on their color content. However, this descriptor can be affected by lighting conditions and the color of the objects that are not the main subject of the image.

GLCM (Gray-Level Co-Occurrence Matrix)

Is used for texture analysis. It works by creating a matrix that represents the occurrences of pairs of pixels with specific gray-level values in each direction.

This descriptor is useful for identifying patterns in an image and can be used for image retrieval and object recognition. It's also robust to changes in lighting and rotation, but it's sensitive to the size and scale of the texture.

HOG (Histogram of Oriented Gradients)

Is used for object detection. It uses edge information and gradient orientations to identify the presence of objects in an image.

The descriptor calculates the gradient magnitude and direction of an image, and then it creates a histogram of the gradient orientations in a local region. This descriptor is useful for object recognition and tracking, but it can be affected by changes in illumination or the rotation of the object.

HSV (Hue, Saturation, Value)

Is also based on the color space, it separates the color information of an image into three channels, hue, saturation, and value. Hue represents the actual color, saturation represents the purity of the color, and value represents the brightness of the color. This descriptor is useful for color-based image retrieval and segmentation, like the BRG descriptor, it can be affected by lighting conditions and the color of the objects that are not the main subject of the image.

SIFT (Scale-Invariant Feature Transform)

Is used for feature detection. It uses local feature information to detect and describe the key points in an image.

The descriptor detects the scale-invariant key points, calculates the gradient orientation at each key point, and creates a histogram of the orientations. This descriptor is useful for image matching, object recognition, and tracking, but it can be affected by changes in the viewpoint, lighting, and scale of the object.

LBP (Local Binary Patterns)

Is used for texture analysis. It uses the local intensity values of an image to create a binary pattern.

The descriptor creates a binary code based on the intensity values of the pixels in a neighborhood, and then it creates a histogram of the binary codes. This descriptor is useful for texture-based image retrieval and object recognition, it's robust to changes in lighting, but it can be affected by changes in the scale and rotation of the texture.

ORB (Oriented FAST and Rotated BRIEF)

descriptor is a combination of the SIFT and BRISK descriptors. It uses feature detection and binary feature description to identify the key points in an image.

The descriptor is based on the FAST algorithm for detecting the key points and the BRIEF descriptor for describing the key points. This descriptor is useful for object recognition and tracking, it's fast and efficient, but it can be affected by changes in viewpoint, lighting, and the scale of the object.

Indexing the database with Deep Learning Models

Indexing a database with deep learning models is a relatively new approach in the field of multimedia retrieval. Deep learning models are neural networks that are trained to recognize patterns and features in multimedia data, such as images, videos, and audio. By using these models to extract features from image files, we can then use them to search and retrieve image files from our database.

The process of indexing a database with deep learning models typically begins with the selection of a pre-trained model or the training of a new model. The selected model is then used to extract features from the multimedia files by feeding them through the network. Once the features have been extracted, they are then used to create an index of the image files, which can be searched and queried later on.

Deep learning models are particularly useful for image indexing, as they can extract more robust and high-level features from the multimedia files. For example, a deep learning model trained on a large dataset of images can be used to extract features such as object detection, semantic segmentation, and image captioning. These features can then be used to search and retrieve images based on their content, such as the presence of a specific object or a specific scene.

VGG16

Is used to extract features from the images by feeding them through the network.

The output of the model, also known as bottleneck features, is extracted from the last fully connected layer of the network, where the features are compressed and contain the most relevant information for image classification.

These features were used to create an index of the images.

VGG models trained on the ImageNet database are particularly useful for image indexing as they can extract robust and high-level features from the images.

Searching using various similarity calculation methods

Searching using various similarity calculation methods is an important step in multimedia retrieval. Different similarity calculation methods were used to compare the features extracted from the images to find the most similar results.

In this project, the database is composed of images of animals, and various similarity calculation methods can be used to search for similar images. However, some methods perform better than others because of the characteristics of our dataset.

When we can search for images of animals based on their visual appearance and use methods such as Euclidean distance and Correlation. These methods are simple and efficient and can provide good results with the images in the database that have similar features. However, these methods might not be suitable when the images have different scales or features.

On the other hand, when searching for images of animals based on their texture or pattern, it is better to use methods such as Chi-squared and Bhattacharyya. These methods are more robust and can provide better results when the images in the database have different features. However, these methods can be computationally expensive.

Intersection and Brute Force Matcher are used for image matching and object recognition tasks. These methods are simple and efficient, but they can be affected by the scale of the features, making them less robust than other methods.

We can use Flann (Fast Library for Approximate Nearest Neighbors), but it will be better to increase our datasets. It's an optimized library that can accelerate the search process, making it efficient for large datasets.

Euclidian, Correlation, Chi-squared, Intersection, Bhattacharyya, Brute Force Matcher and Flann are the similarity calculation function that we used for searching similar images.

Each function has its own unique characteristics and is best suited for certain types of image analysis; we will describe their different characteristics very shortly

Euclidean distance

This method calculates the straight-line distance between two feature vectors. It is simple to calculate, and it's widely used in image retrieval tasks. However, it doesn't consider the correlation between features, which can be a problem when the features have different scales.

Correlation

This method calculates the correlation between two feature vectors. It's useful for finding patterns in the data, and it's widely used in image retrieval tasks. However, it can be affected by the scale of the features, which can make it less robust than other methods.

Chi-squared

This method calculates the similarity between two feature vectors by comparing the distribution of their values. It's a robust method, as it's less affected by the scale of the features, but it can be computationally expensive. It's mainly used in image recognition tasks.

Intersection

This method calculates the similarity between two feature vectors by measuring the overlap between their values. It's a simple and efficient method, but it can be affected by the scale of the features, making it less robust than other methods. It's mainly used in image matching and object recognition tasks.

Bhattacharyya

This method calculates the similarity between two feature vectors by measuring the overlap between their probability distributions. It's a robust method, as it's less affected by the scale of the features, but it can be computationally expensive. It's mainly used in image recognition tasks.

Brute Force Matcher

This method compares each feature vector in the database with the query feature vector. It's useful for small datasets, but it can be computationally expensive for large datasets. It's mainly used in object recognition and image-matching tasks.

Flann (Fast Library for Approximate Nearest Neighbors)

It's an optimized library for approximate nearest-neighbor search, it uses tree-based data structures to accelerate the search process, making it efficient for large datasets. It's mainly used in object recognition and image-matching tasks

Calculation of metrics

In multimedia retrieval, recall, precision, average precision, mean average precision and r-precision are commonly used metrics to evaluate the performance of a retrieval system.

They help to measure the proportion of relevant items retrieved by the system, the proportion of retrieved items that are relevant, the ranking of the retrieved items, the overall performance of a retrieval system and the performance of a retrieval system for a specific level of recall. These metrics are useful for analyzing the performance of retrieval systems and can help to identify areas for improvement.

Recall (R)

measures the proportion of relevant items in the dataset that are successfully retrieved by the system. It is calculated as the number of relevant items retrieved by the system divided by the total number of relevant items in the dataset.

Precision (P)

measures the proportion of retrieved items that are relevant. It is calculated as the number of relevant items retrieved by the system divided by the total number of items retrieved by the system.

Average Precision (AP)

is an extension of precision that considers the ranking of the retrieved items. It is calculated as the average precision at each relevant item in the retrieval results.

Mean Average Precision (MAP)

is the mean of the average precision over multiple queries. It is a measure of the overall performance of a retrieval system.

R-Precision

is a measure of the performance of a retrieval system for a specific level of recall. It is calculated as the precision at the first R-recall level.

Table 1: Indexing and Search Performance Metrics

Your best descriptors	Name of descriptor(s)	Indexing time(s)	Indexing Size(MB)	Average search time per frame(s)
Descripteur N° 01	BGR	128,26	48,7	0,05
Descripteur N° 02	HSV	113,29	48,7	0,04
Descripteur N° 03	SIFT	997,83	8130	0,37
Descripteur N° 04	ORB	253,64	970	0,10
Descripteur N° 05	GLCM	253,86	3,04	0,10
Descripteur N° 06	LBP	281,37	406	0,11
Descripteur N° 07	HOG	209,19	386	0,08

Table 2: Search Engine Accuracy Metrics

Query index	R		P		AP		MaP	
	Top50	Top100	Top50	Top100	Top50	Top100	Top50	Top100
R1 - 0_5_araignees_tarantula_795	0,41	0,32	0,74	0,69	0,82	0,79	0,71	0,68
R2 - 0_4_araignees_gardenspider_631	0,62	0,49	0,43	0,37	0,43	0,4		
R3 - 0_1_araignees_wolfspider_259	0,43	0,28	0,33	0,3	0,39	0,35		
R4 - 1_0_chiens_Siberianhusky_849	0,41	0,31	0,89	0,81	0,92	0,87		
R5 - 1_3_chiens_Chihuahua_1315	0,32	0,24	0,81	0,76	0,83	0,8		
R6 - 1_1_chiens_Labradorretriever_1054	0,45	0,33	0,6	0,54	0,61	0,58		
R7 - 2_2_oiseaux_greatgreyowl_2092	0,26	0,19	0,95	0,9	0,94	0,91		
R8 - 2_4_oiseaux_robin_2359	0,54	0,31	0,64	0,58	0,6	0,59		
R9 - 2_3_oiseaux_bluejay_2232	0,47	0,29	0,82	0,77	0,83	0,81		

Part 02: hosting the application on Cloud resource

The goal of this part has to host the media search application (from part 1) on a Cloud resource to offer a service in the form of Software As A Service "SAAS"

To achieve this goal, the application was containerized using Docker and the web framework Flask was used to develop the application.

Docker is a containerization platform that allows for the development and deployment of applications in a portable and consistent environment. This was useful in this project as it allowed for the easy deployment of the application on the cloud resource.

Flask is a lightweight web framework for Python. It was used to develop the application's user interface, allowing users to easily search and access multimedia data.

The use of Docker and Flask in this project allowed for the efficient development and deployment of the multimedia search application on a cloud resource. As a result, users can access the application as a SaaS offering and easily search and access multimedia data. Additionally, some print screens of the application were included in the report to show the functionality and user interface of the application.

Local "characteristic extraction" indexing

In this project, the characteristic extraction for indexing the images in the database was performed locally due to the limited performance of the virtual machine used. The virtual machine used did not have a GPU, which is necessary for the efficient processing of the large dataset of images. Therefore, the indexing phase was not hosted on a cloud resource but rather on a local machine.

The images were first pre-processed, and then feature extraction was applied using several descriptors like GLCM, HOG, HSV, SIFT, and some deep learning models, to extract the characteristics of the images that would be used to create the index. The resulting index was then transferred to the virtual machine via Secure Copy Protocol (SCP) to be used in the search phase of the application.

This approach allowed for efficient and accurate indexing of the images, despite the limitations of the virtual machine. However, it should be noted that this process could be time-consuming and could take a significant amount of storage space.

Testing and configuring the cloud search application

The application was configured on a virtual machine to test it on a cloud resource.

To do this, several steps were necessary to install and configure the necessary software and dependencies on the virtual machine.

First, the virtual machine was set up with the Ubuntu operating system, and the necessary libraries and dependencies were installed. This included installing Python, the Flask web framework, and other libraries used in the application, such as OpenCV, NumPy, etc.

Next, the image database and the index created locally were transferred to the virtual machine via Secure Copy Protocol (SCP) and properly configured to be used by the application.

Additionally, the cloud resource was set up and configured to host the application, which involved opening the necessary ports.

Finally, the application was tested on the cloud resource to ensure that it was running correctly and that all the necessary configurations had been made.

Generation of the Docker image

A Docker image was generated to group together the functionalities of our application.

For that, a Dockerfile was created, which contains the instructions needed to run the application. The Dockerfile defines the base image to be used, then, through the requirements.txt file, specifies the necessary dependencies and libraries to be installed, such as Flask, OpenCV, NumPy, etc.

Next, it copied the application code and the indexing data to the image.

The Dockerfile also defines the working directory for the application, the environment variables, and the command to run the application when the container is launched.

The Docker image was built by executing the command "docker build" on the command line, giving the path to the Dockerfile.

This process created an image that contains all the necessary components and dependencies to run the application.

Development of a web page to facilitate access to the SAAS service

The web page was developed using the Flask web framework to facilitate access to the Software as a Service (SaaS) multimedia search application.

The web page was designed to provide an intuitive and user-friendly interface for accessing the SaaS service. It allows users to perform searches easily.

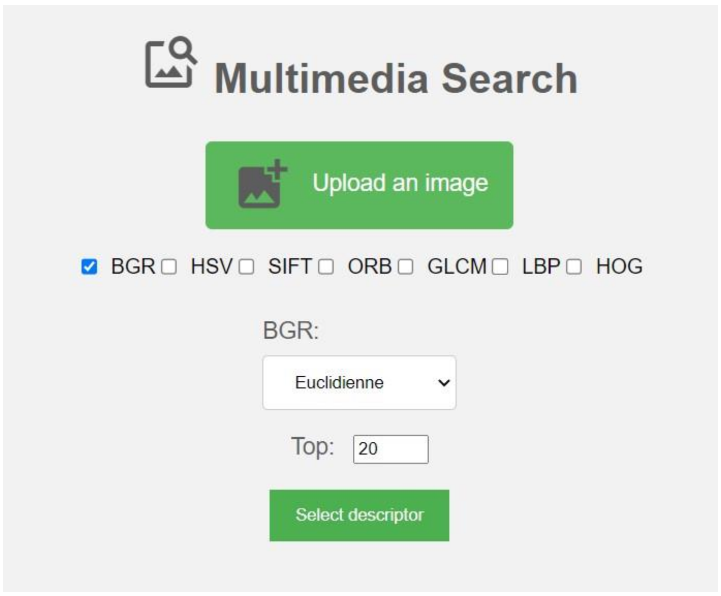
The web page provides a description and features of the multimedia search application.

The web page was designed with a user-friendly interface that allows users to launch the search application using buttons, labels, and other UI elements.

The search results are displayed in an organized way, displaying the top X similar images as well as R/P curves, which provide a measure of the performance of the retrieval system.

The web page allows users to easily navigate and access the multimedia search application.

Fig1: Flask App select descriptor

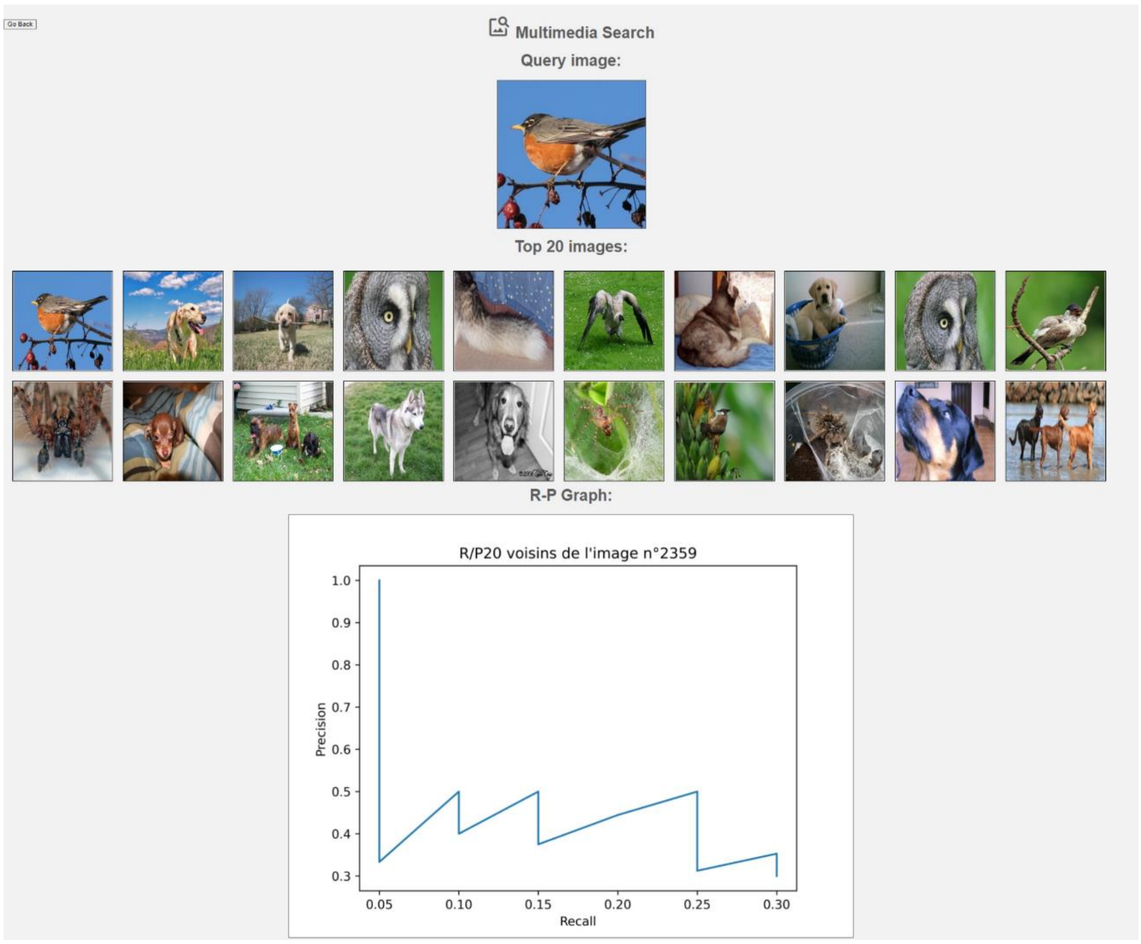


[✓ BGR] ☐ HSV ☐ SIFT ☐ ORB ☐ GLCM ☐ LBP ☐ HOG

BGR: Euclidienne

Top: 20

Fig2: Flask App results



Conclusions

Indexing a database with descriptors is an essential phase in the multimedia retrieval process. It enables us to extract relevant characteristics from multimedia files and develop a searchable index, allowing us to search and retrieve multimedia files from a database effectively.

On the other hand, indexing a database with deep learning models is a promising approach in the field of multimedia retrieval. It allows us to extract more robust and high-level features from the multimedia files, making it possible to efficiently search and retrieve image files from a large database. However, it is computationally intensive and requires a large dataset for training the models.

The choice of the descriptor is determined by the specific application and requirements, each descriptor has distinct properties and is better suited for different types of image analysis.

Our database contains photos of animals that change depending on the species, the surroundings, and the angle of the image, so the performance of each descriptor may also vary.

Different similarity calculation methods can be used to search for similar images in a database. Some methods, such as Euclidean distance and Correlation, are simple and efficient, but they don't take into account the correlation between features. Other methods, such as Chi-squared and Bhattacharyya, are more robust, but they can be computationally expensive.

It's important to choose the right similarity calculation method depending on the size of the dataset, and the computational resources available.

This multimedia retrieval system was developed on a cloud resource utilizing multiple descriptors and deep learning models, Docker, and flask, and provides a SaaS service. The system was tested using several similarity calculation methods, and the results were analyzed to determine which one performed better. This project demonstrates the potential of cloud computing and multimedia retrieval in providing efficient and accurate access to multimedia data.

References

Deploying your deep learning model using Flask and Docker:

Web app for image labelling