Reverse Image Querying

Janhavi Anap[1], Prathamesh Hambar[2], Tejas Sheth[3], Het Patel[4], Rushikesh Nikam[5]

[1-4]Students, [5]Professor, Department of Computer Engineering, AP Shah Institute of Technology. Thane West

[1]anapjanhavi@gmail.com

[2]pphambar@gmail.com

[3]tejas.sheth04@gmail.com

[4]hetpokar@gmail.com

[5]rrnikam@apsit.edu.in

***Abstract*— As days go by, the volume of graphic i.e. visual data keeps increasing exponentially. This presents a more recent and complex set of problems for handling and making use of such vast, disorganised and unregulated data. Multiple organisations have launched many commercial products to tap into this market. There are multiple methods and functions for reverse image querying but most of them are either incomplete or compute-intensive to be a viable option for smaller use cases. In this study, we have implemented and improvised a rudimentary yet efficient solution by using the cosine similarity model to extract similar images for the given input image from the user. The model used is explained comprehensively together with visual representations. We have also explored multiple use cases that can be used as commercial products. The resulting web application can be utilised to act as a reverse image search engine as a standalone application or can be embedded as a sub-module in a larger application.**

***Keywords*—– image processing, image compression, cosine similarity model, reverse image search, colour feature extraction**

I. Introduction

Information forms the basis of our knowledge. Over the years, we have relied on books, newspapers, and other physical media. They were all stored in libraries or any other educational institutions as limited physical copies. With digitisation, the amount of information that can be stored in a limited space keeps growing exponentially. Then came the era of the World Wide Web (WWW). This enabled a whole new set of information that was generated in huge amounts every day. But the most important aspect of this was how rapidly a lot of information was being shared.

One of the first conceptualisations of a search engine was given by Vannevar Bush in 1939 and it was elaborated on and published in July 1945 in the article "As We May Think" (Bush 2022) of The Atlantic magazine.

He envisioned Memex (a portmanteau for memory and expansion), a hypothetical electromechanical device in which individuals would compress and store all of their books, records, and communications. This concept highly influenced the development of early hypertext systems, which eventually gave rise to the World Wide Web and further complex technologies based on this.

With the coming of WWW, the surge of usage of hyperlinks and studies on link analysis pioneered crucial components of search engines through algorithms such as Hyper Search and Page Rank. Soon, web crawlers were being used for web indexing.

Post the Dotcom bubble, multiple social media websites started to gain popularity with Facebook, Reddit, Pinterest, Instagram, etc. These websites heavily focused on media sharing more prominently on image sharing. As the internet became more and more accessible, data generation grew steadily. More than a billion photos are uploaded every day and this keeps on incessantly increasing.

This posed a challenge to the developers to make image retrieval systems to cater to such a high influx of images accurately. An image retrieval system is a system used for browsing, searching and retrieving images from a large database of digital images. Image search is a specialised data search used to find images. To search for images, a user may provide query terms such as keyword, image file/link, or click on some image, and the system will return images "similar" to the query. The similarity used for search criteria could be meta tags, colour distribution in images, region/shape attributes, etc.

Its types are:

• Image Meta Search: Usually, metadata is added such as title, captioning, keywords, description, etc. to the images to retrieve them later using annotation words.

• Image Collection Exploration: It is a paradigm to explore large digital image repositories.

• Content-based Image Retrieval: Instead of textual description, CBIR utilises Computer Vision techniques to index images based on their contents/features such as colour, texture, shape, objects, etc.

To perform CBIR, there are multiple techniques based on the type of user query.

The prominent ones are

• Query By Example (QBE): This includes providing an example to the CBIR to base upon its search. Semantic Retrieval: A user usually makes an open-ended request which is difficult to process. The CBIR then extracts the features such as colour, texture, etc. and finds similar images containing those features.

• Relevant Feedback: In this, the user progressively refines the search results by marking images in the results as ”relevant”, ”not relevant”, or ”neutral” to the search query, then repeating the search with the new information.

• Iterative/Machine Learning: Convolutional Neural Networks among others are used to find similar images.

Modern-day search engines are highly complex and use parts of multiple techniques or make their own state-of-the-art algorithms to provide accurate results with the possible latency.

In this project, we will be creating a rudimentary CBIR-based QBE search engine with a dataset. Similar images will be queried using the cosine similarity model. A basic interface will be made to facilitate the search by the users.

II. Literature Survey

There are multiple products for identifying the most similar image from a given large image database. Reference [1] states about multiple shortcomings of the current software, databases being disorganised, having no regulation, and the room for improvement for the similarity models. It compares the multiple similarity models such as Cosine similarity, Pearson similarity, Correlation distance, Chebyshev distance, Manhattan distance, and Euclidean distance. The paper assumes that we are looking for a similar image, not the same image; the features of vectors which are in the same direction can be recognized similarly. The specific distance is not really important. Other models have a much lower accuracy as compared to the cosine similarity model.

This proves to be quite efficient, but it also presents a critical flaw. This model is dependent on any other feature extraction model that is used before it. Thus, any errors or misinterpretation of a feature could give dramatically different results. Another limitation of such a system is that it can only be used for object-based applications. It cannot be used to do a biometric scan for a human face i.e. it cannot distinguish between slight differences between features if not heavily trained and tested. Thus, it can be improved to overcome its flaws.

Image retrieval is the technique of retrieving similar pictures from an image database. Many search engines providing image retrieval are based on text-based or keyword-based approaches. They take text as an input and give similar images as an output. They also tend to provide results based on the caption using Natural Language Processing. It is also tedious to manually annotate keywords for pictures on the net to retrieve actual and similar images. But it becomes very troublesome to interpret the user's intention only based on a few keywords; hence the search results turn out ambiguous and unsatisfactory. Thus, to improve the results, the authors in [2] used reverse image search engines. They used content-based image retrieval, which outputs images based on features like colour, texture, shape, spatial layout, etc. The reverse image search engines classify their images based on visual characteristics like colour, texture, and form, which are extracted from the image itself. Some popular reverse search engines include Google, TinEye, and Yandex. The size of their data set is 11.94 billion, 9.14 billion, and 5.6 billion, respectively. Google has the most abundant image information and uses various algorithms considering various attributes to retrieve the output.

To search for similar images, right now, Google Image Search is the most popular software application. The authors in [3] suggest a few methods clubbed together to achieve image enhancements. The use of features to represent the image content along with its average R-G-B cluster, which is invariant against translation and rotation, in the form of groups to display the images. Since the clusters contain mainly visually similar images, the results obtained are a great starting point.

The search for many words is ambiguous and results in varied output images. Whereas, we almost always obtain two groups as a result, even for words with lesser ambiguity. We have presented an approach using image processing and image retrieval methods, to facilitate the user to meet his search requirements faster.

Reference [3] presents methods like Pearson similarity, correlation distance, etc. to improvise image-based searching in data sets for (most) similar images. To obtain more precise results they use a variety of different algorithms to compare cluster results. Using the proposed features in other applications like image retrieval and copyright infringements, we hope to gain further information on how to improve the results.

The algorithm that the authors in [4] present for searching and retrieving images separates Red, Green and Blue colour planes and calculates the average of all row means and all column means of each plane. To form a feature vector, the features of all three planes are combined. Feature vectors of all images present in the database once calculated are stored in the feature database. The Euclidean distance between the query image's feature vector and the feature database is calculated by the system. Similar images are fetched from the database on the basis of the low value of Euclidean distance. The Low value of Euclidean distance indicates higher relevance to the query image. The algorithm produced better results with the smaller data set. The size of the data set and the number of different classes affected the relevancy of retrieved images.

In various fields, large collections of digital images are being created. Usually, the only way of searching through these collections is through keywords. Reference [5] surveyed technical aspects of various content-based image retrieval systems and found that out of 56 systems that were reviewed, 46 systems used colour feature extraction, 38 used texture, 29 used shape, and 5 used face detection. The easier the feature is to extract, the easier it is to implement it in the system, and the easier it is to use that feature class. Most of the systems use colour and texture feature extraction since they are found to be effective. But retrieval of images based on texture doesn't always yield images with similar texture unless the data set contains most of a particular texture.

The yielded result also depends on the kind of data set used. A general search in a biassed data set would give unsatisfactory results and using a small and insufficient data set would yield irrelevant results. Hence having an equally diverse data set is also very important.

III. Experimental Setup

A. Software Requirements

Python: General Programming Language used to code the model, includes several libraries.
• Python libraries: numpy, matplotlib, PIL (Python Imaging Library), pandas, streamlit, os.
• VS Code/Anaconda: IDE to run, train and test the ML model. Plenty of extensions, open-source, cross-platform support,
• Git and GitHub: Version Control System used for collaboration.
• Streamlit: Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps.
• Google Workspace: Project Management Tool to improve team coordination and file sharing.

B. Hardware Requirements

• CPU: 1.8 GHz or faster 64-bit processor; Quad-core or better recommended.
• RAM: Minimum of 4GB of ram.
• Storage: 4GB of free hard disk space

IV. Methods And Process

The method of reverse image querying is to retrieve images similar to the input image by calculating the cosine distances. Refer Fig. 1 and Fig. 2.

The steps of the algorithm are given below:

Step1: The input is a colour image in R-G-B colour space.
Step2: Load the image dataset's pickle file into a data frame containing the images and their average R-G-B values.
Step3: Get the average R-G-B values of the input image.
Step4: Construct a feature vector that will represent the image and the average R-G-B values.

```
image = Image.open(dir + "/" + img)
img_array = np.array(image)
red = np.average(img_array[:, :, 0])
green= np.average(img_array[:, :, 1])
blue= np.average(img_array[:, :, 2])
```

Step5: Using the average R-G-B values, calculate and store the cosine distances between the input query image and each image in the data frame. Cosine Similarity and Cosine Distance is given by

$$S_C(A,B) := \cos(\theta) = \frac{A.B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \cdots (1)$$

$$\text{cosine distance} = D_c(A,B) := 1 - S_c(A,B) \ldots(2)$$

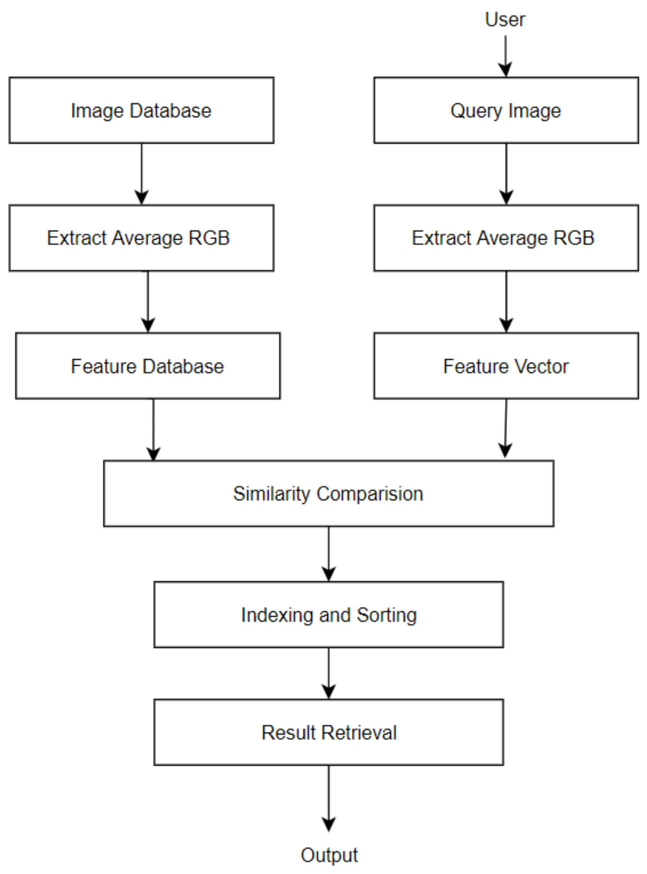


Fig. 1 Architecture Diagram

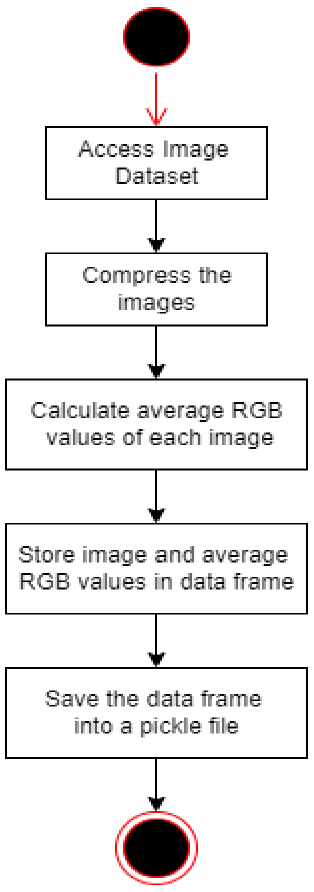

Access Image Dataset
Compress the images
Calculate average RGB values of each image
Store image and average RGB values in data frame
Save the data frame into a pickle file

Fig. 2 Pre-processing Activity Diagram

```
cosine_dist=[]
idx=0
for i in range(len(df)):
        temp=df.iloc[i,-3:]
        temp=np.array(temp).reshape(1,-1)
        dist=cosine(query_feature[-3:],temp)
        cosine_dist.append([dist, idx])
        idx += 1
```

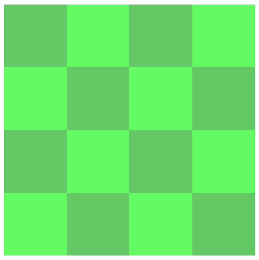
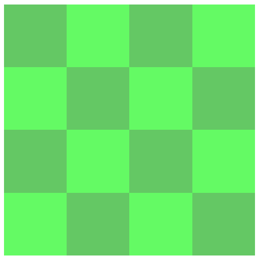
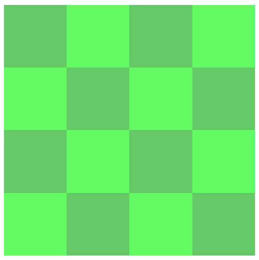
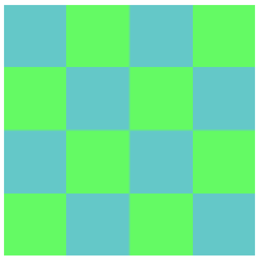
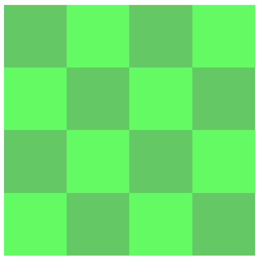
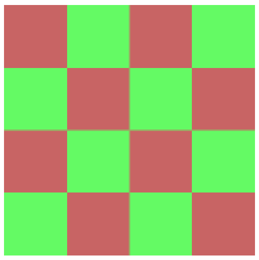
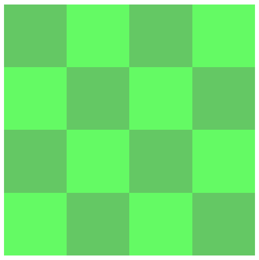
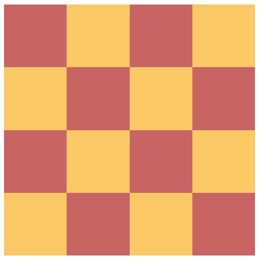
Step6: Sort the distances in ascending order.

```
cosine_dist.sort()
result = []
for dist, idx in cosine_dist:
        result.append(idx)
```

Step7: Retrieve and display the first few images having the least cosine distance.

TABLE I

EXAMPLE

Query Image	**Sample Image**	**Cosine Distance** [a b]
 **A = [100, 225, 100]**	 **B = [100, 225, 100]**	A.B = (100*100) + (225*225) + (100*100) = 70625 $\lvert A\rvert = \sqrt{100^2+225^2+100^2} = 265.75$ $\lvert B\rvert = \sqrt{100^2+225^2+100^2} = 265.75$ $S_c(A, B) = 70625/(265.75*265.75) = 1$ $\mathbf{D_c(A, B) = 1 - 1 = 0}$
 **A = [100, 225, 100]**	 **B = [100, 225, 150]**	A.B = (100*100) + (225*225) + (100*150) = 75625 $\lvert A\rvert = \sqrt{100^2+225^2+100^2} = 265.75$ $\lvert B\rvert = \sqrt{100^2+225^2+150^2} = 288.31$ $S_c(A, B) = 75625/(265.75*288.31) = 0.987$ $\mathbf{D_c(A, B) = 1 - 0.987 = 0.013}$
 **A = [100, 225, 100]**	 **B = [150, 175, 100]**	A.B = (100*150) + (225*175) + (100*150) = 64375 $\lvert A\rvert = \sqrt{100^2+225^2+100^2} = 265.75$ $\lvert B\rvert = \sqrt{150^2+175^2+150^2} = 251.25$ $S_c(A, B) = 64375/(265.75*251.25) = 0.964$ $\mathbf{D_c(A, B) = 1 - 0.964 = 0.036}$
 **A = [100, 225, 100]**	 **B = [225, 150, 100]**	A.B = (100*225) + (225*150) + (100*100) = 66250 $\lvert A\rvert = \sqrt{100^2+225^2+100^2} = 265.75$ $\lvert B\rvert = \sqrt{225^2+150^2+100^2} = 288.31$ $S_c(A, B) = 66250/(265.75*288.31) = 0.865$ $\mathbf{D_c(A, B) = 1 - 0.865 = 0.135}$

[a] Refer (1) [b] Refer (2)

V. Results

Our final decision was to use the cosine similarity model, we came to the consensus after thorough research on the cosine similarity model and found out that it was, by far, the most effective model to fetch the most similar images from the given database. We focused on retrieving similar images based on their average R-G-B colour attributes. We managed to achieve great results, as seen in Fig. 3 using simple yet highly effective algorithms. Using cosine distances (accuracy 56.76%) proved to be the most efficient method to calculate similarity over chebyshev (accuracy 55.58%) and euclidean (accuracy 53.23%) by providing better and faster results. One of the outstanding problems was the ability to say definitively if the two products were not considered to be similar. This is a difficult problem to solve because there will be times when we need to search for an image that we do not necessarily have any similarities with, but still want to try and find the closest match. It can be seen in Table. 1 that as we increase the deviation of the Red, Green, and Blue values of the query image, its cosine distance from the original image increases. Hence, the cosine distances between the query and sample image determine how similar the two images are.

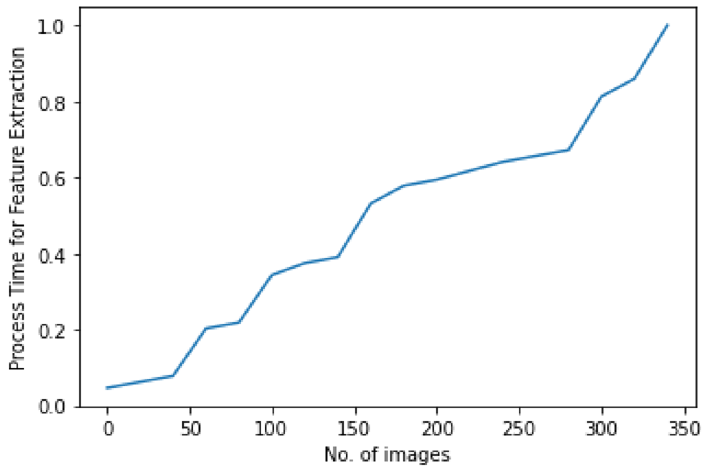


Fig. 3 Processing Time for Feature Extraction

VI. Conclusions

According to our results, using the best similarity model we choose, there has been a significant increase in accuracy, based on the colour attributes. We managed to achieve great results using simple yet highly effective algorithms. We can further improve the results of the model with the help of better hardware or accelerated computing or high-performance computing techniques like multi-threading, and many more. This model can be implemented in a number of sectors including fashion, design, automobile, and Fast-moving consumer goods (FMCG). Moreover, it can be utilised to realise manipulated versions of any image, and protect against unauthorised reuse of published images.

References

[1] Chen, Ye. (2020). Exploring the Impact of Similarity Model to Identify the Most Similar Image from a Large Image Database. Journal of Physics: Conference Series. 1693. 012139. 10.1088/1742-6596/1693/1/012139.

[2] Adrakatti, Arun and Wodeyar, R. and K.R, Mulla. (2016). Search by Image: A Novel Approach to Content Based Image Retrieval System. International Journal of Library Science. 14. 41-47.

[3] Deselaers, Thomas and Keysers, Daniel and Ney, Hermann. (2003). Clustering visually similar images to improve image search engines.

[4] Kekre, Hemant and Mishra, Dhirendra and Narula, Ms and Shah, Vidhi. (2011). COLOR FEATURE EXTRACTION FOR CBIR. International Journal of Engineering Science and Technology.

[5] Veltkamp, Remco and Tanase, Mirela. (2000). Content-Based Image Retrieval Systems: A Survey. Technical report, Utrecht University.

[6] Luo, Bo and Wang, Xiaogang and Tang, Xiaoou. (2003). World Wide Web Based Image Search Engine Using Text and Image Content Features. Proc SPIE. 10.1117/12.476329.

[7] Brin, S., and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems, 30(1-7), 107–117.

[8] Kekre, Hemant and Mishra, Dhirendra. (2010). CBIR using Upper Six FFT Sectors of Color Images for Feature Vector Generation. International Journal of Engineering and Technology. 2.

[9] Gudivada, V. N., and Raghavan, V. V. (1995). Content based image retrieval systems. Computer, 28(9), 18–22. doi:10.1109/2.410145