CS 5330 Project-2

Saideep Arikontham

Title: Exploring Video Filters and Real-Time Effects Using OpenCV

Project Overview

This project focuses on developing different Content-Based Image Retrieval (CBIR) techniques and different ways to evaluate the results. These techniques help in finding images similar to a given target image from a database. The project involves techniques like color histograms, texture histograms, different combinations of these histograms, depth information and deep neural network embeddings for generating feature vectors for images.

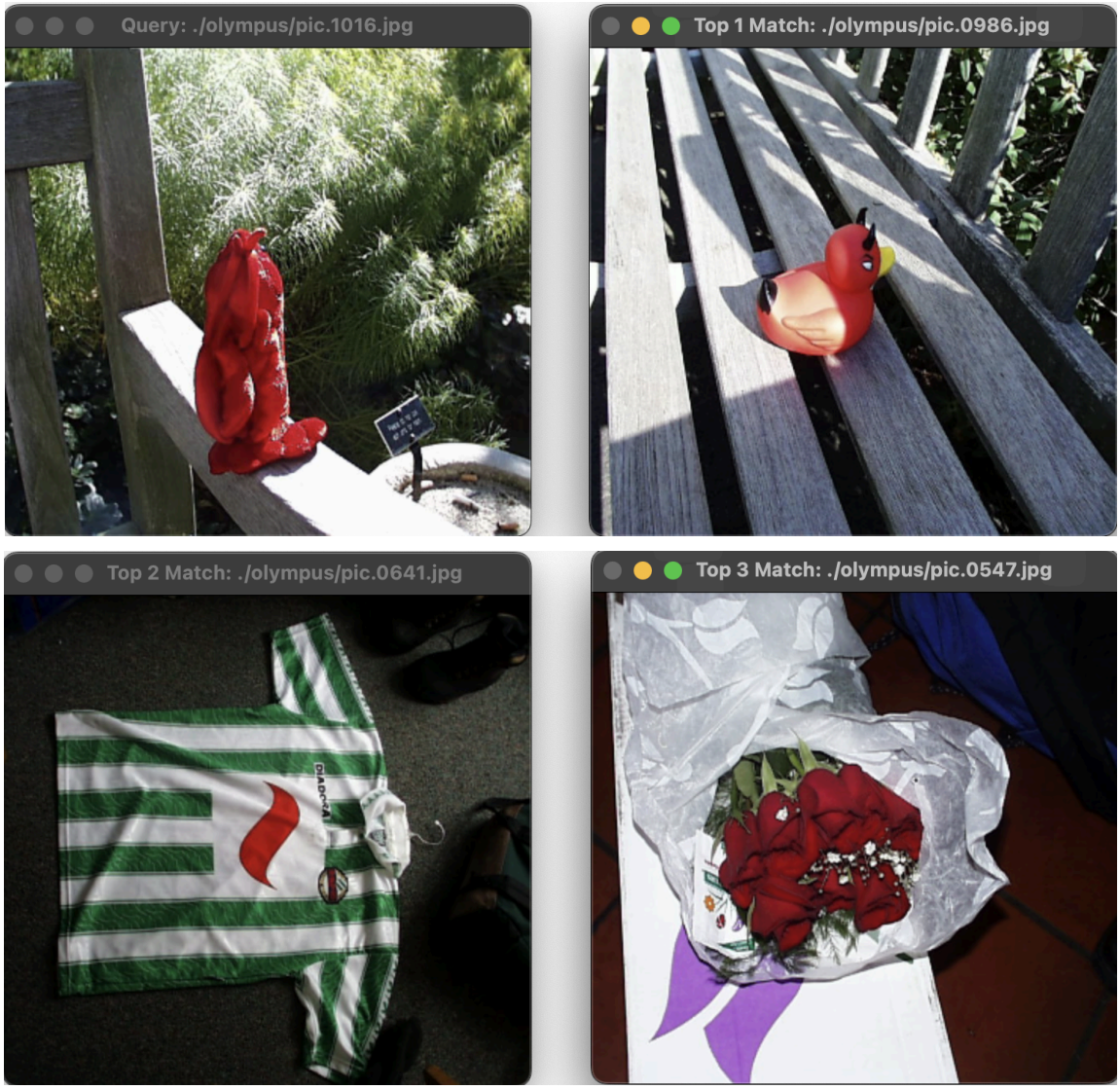
The exact process of generating feature vectors and identifying match is as listed below:

1. Generating feature vectors for all images in the database.
2. Select a target image that we want to use as a query and generate feature vectors for this image.
3. Compare the feature vectors of the target image to those in the image database.
4. Display the most similar images in ranked order.

Before extracting the feature set from any image, we will resize it to a fixed 640x512 pixel size to ensure uniformity. We will understand the strengths and weaknesses of different techniques along with the interpretation of different histogram evaluation techniques. To save time, we first calculate feature vectors for all images in the database. Then after that, we use the selected feature vector file and query image to find the top matches.

Directed Tasks and required images

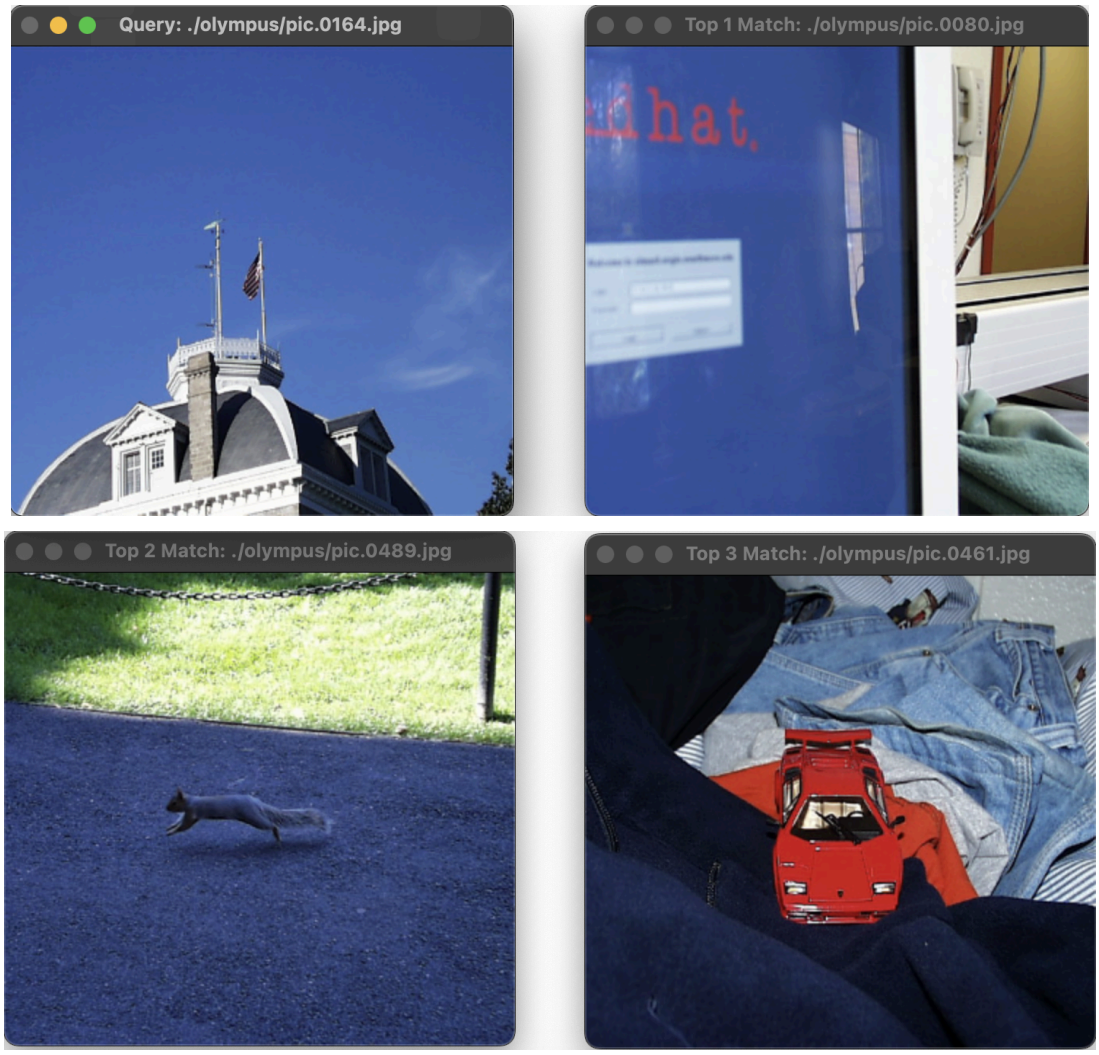
1. **Baseline Matching (Req. result 1):** In this method, we directly try to match the color channels of each pixel for query image and target. For this task, we used a 7x7 pixel grid from the center. Therefore, the feature vector for an image would have 7*7*3 dimensions (7*7 pixels and 3 channels per pixel). Sum of squared distances is used as an evaluation metric (sum of square of difference between each dimension is evaluated). For the image *pic.1016.jpg,* below are the top 3 matches identified.



We can see that the center pixels are red in the query image and the top 3 matches reflect the same.

2. **Histogram matching (Req. result 2):** In Histogram matching, we use a simple RG histogram. An RG histogram represents the color distribution of an image using only the red (R) and green (G) channels, ignoring the blue (B) component. It is also called as RG chromaticity used in chromaticity based color analysis focusing primarily on color composition. The RG chromaticity of each pixel is calculated, mapped to the respective histogram bins performing increment operation. We are using a histogram with 16 bins and therefore, the resulting feature vectors would have 16*16 dimensions. Histogram

intersection is used as a metric (minimum value at each dimension is identified to get the intersection). For the image *pic.0164.jpg,* below are the top 3 matches identified.



- We can see that the query image has a lot of blue, grey and shade of white. Using the histogram matching with RG chromaticity, we can see that the top matches all have the shades of these three colors.

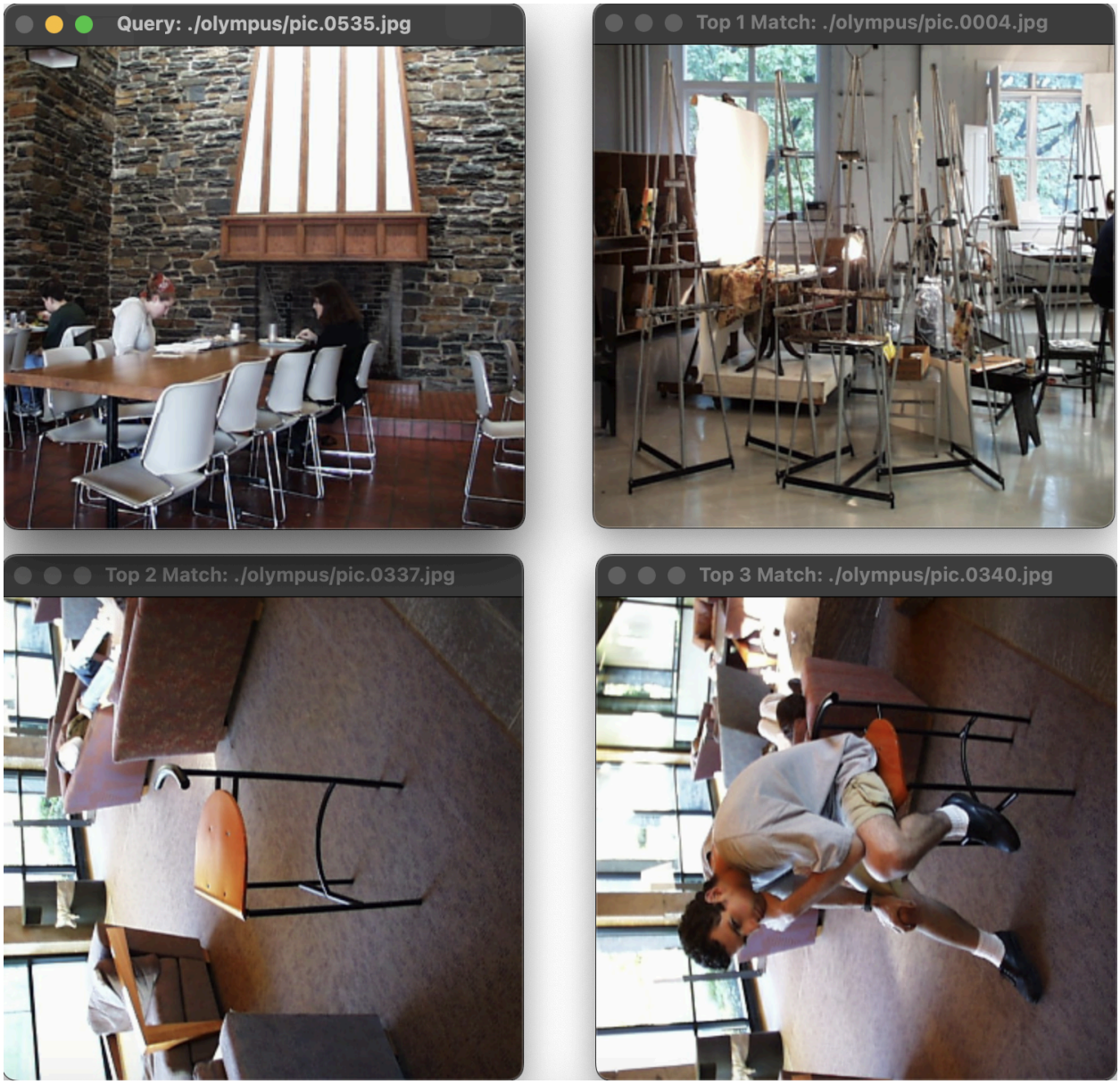
3. **Multi-Histogram matching (Req. result 3):** In Multi-Histogram matching, we use a RGB histogram. An RGB histogram is a 3D histogram that represents the color distribution of an image by counting pixel intensities separately for the Red, Green, and Blue channels. We are using two histograms with 8 bins each and therefore, the resulting feature vectors would have 8*8*8 dimensions each. One histogram is for the entire image and the other histogram is focused on the middle rectangle of the histogram. Histogram intersection is used as a metric (minimum value at each dimension

is identified to get the intersection). The resulting two histogram intersection scores are weighted to identify a final score. For the image *pic.0274.jpg,* below are the top 3 matches identified.

- We can see that the color palette for all the images is almost the same. The sky, the green from trees and the white part (clouds or buildings) might be prominent in determining these matches.

4. **Texture and Color (Req. Image 4):** In this task, we use the combination of the whole image RGB Histogram and the whole image texture histogram. I have chosen the sobel magnitude as my texture and calculated histogram for it as part of texture histogram. This allows not only to compare color components but also takes the underlying texture into consideration. For this task, I have used chi-squared distance as the metric (dissimilarity between two distributions by summing the squared differences between

corresponding values, normalized by the expected values). For the image *pic.0535.jpg,* below are the top 3 matches identified.

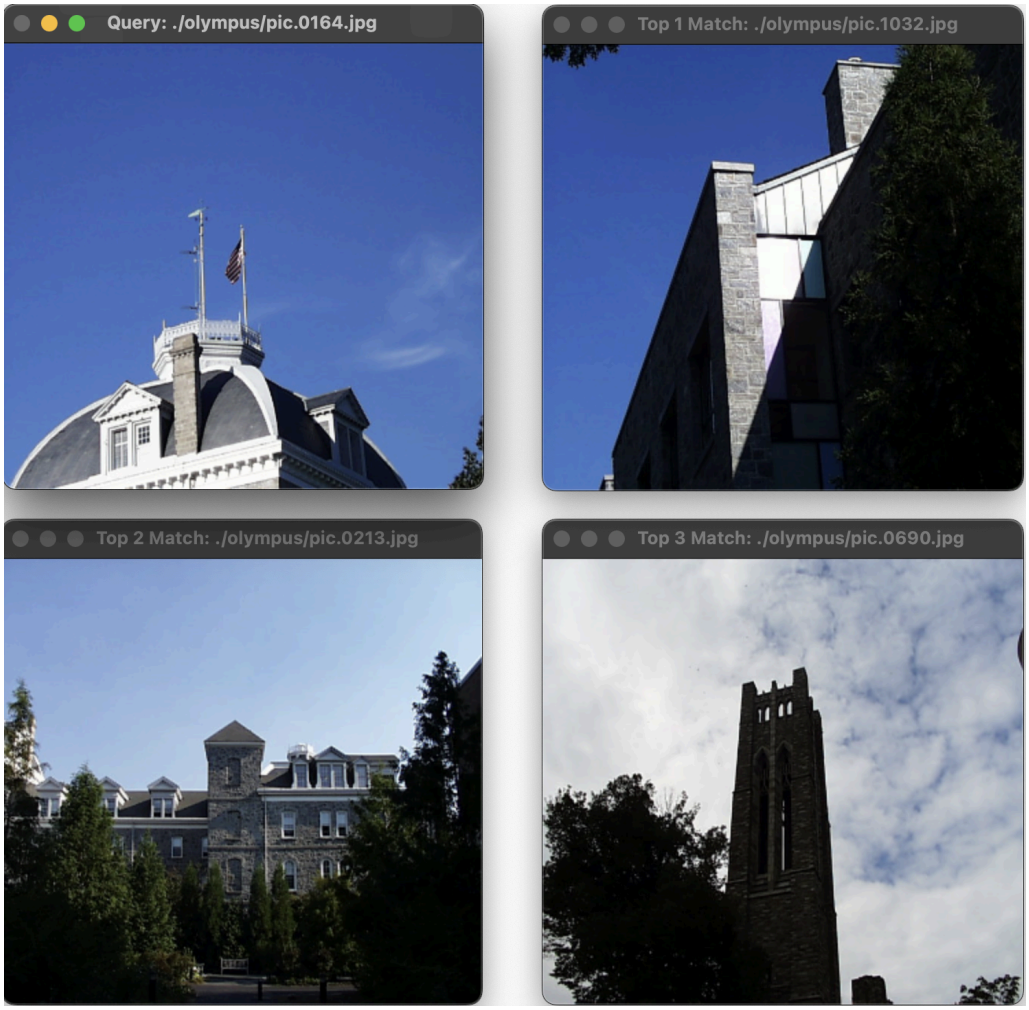


- We can see that previously, we only relied on color components of the images to build histograms and compare them. This reflected in the query match results where color components are a major part. Task one was entirely comparing the color channel values. Task two and three used 2D RG histogram and 3D RGB histogram which returned the best possible images based on color distribution. But here, we can see that there are lot of vertical and horizontal lines in the query image which would have weight in the texture histogram. We can see the top matches as well. Not only the color palette matches, but the texture seems comparable with a lot of vertical and horizontal lines. Including texture

adds an additional layer of filtering to identify the top matches, solidifying the content based image retrieval.

5. **Deep Network Embeddings (Req. result 5):** Deep network embeddings are robust and reliable feature vectors extracted from a pre-trained convolutional neural network (CNN) such as ResNet18. These embeddings capture high-level patterns and semantic information from images. To obtain them, we select a specific layer whose output serves as the feature vectors. These embeddings are more discriminative than histograms and textures, making them effective for image retrieval and similarity tasks.. For this task the feature vectors have 512 dimensions and the feature vector file is already given. We use cosine distance (1 - cosine similarity) as the performance metric for this task (computes dissimilarity by measuring the cosine of the angle between two vectors). Below are the top 3 matches for image *pic.0893.jpg.*

- Below are the top 3 matches for the image *pic.0164.jpg.*



- We can see that the top matches do not have a consistent color palette but have similar objects/ patterns.

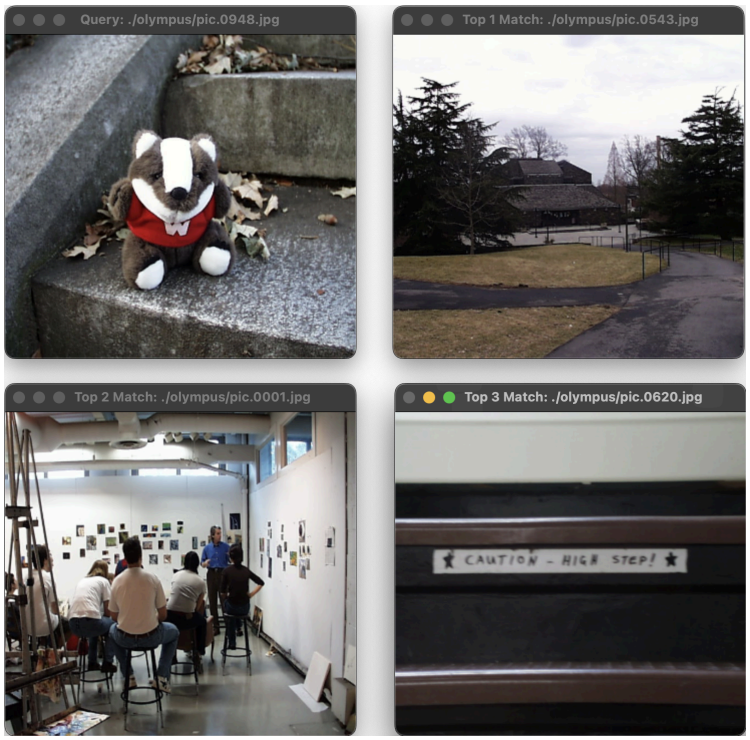
6. **Compare DNN Embeddings and Classic Features:** As explained previously, the deep neural networks are almost always better than classic features like histograms. This is because deep neural networks are trained to identify objects and patterns, specifically image classification tasks. ResNet18 is a deep network with 18 convolutional layers and is pre-trained on one million images. So, it is a robust neural network architecture which is bound to identify patterns and objects. But the classic features mostly rely on the color palette and texture (when texture histograms are used) but fail to identify objects perfectly. Below are the top 3 matches for using multiple RGB histograms and deep neural networks respectively.

- For image *pic.1074.jpg:*
    - Using Muti-Histograms:

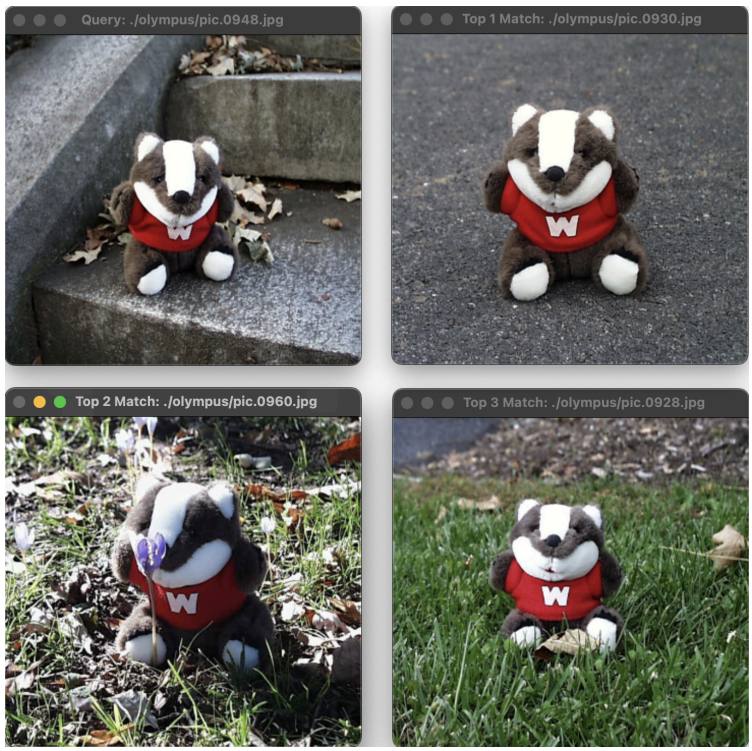
- Using Deep neural networks:

- Top 3 matches for *pic.0948.jpg:*

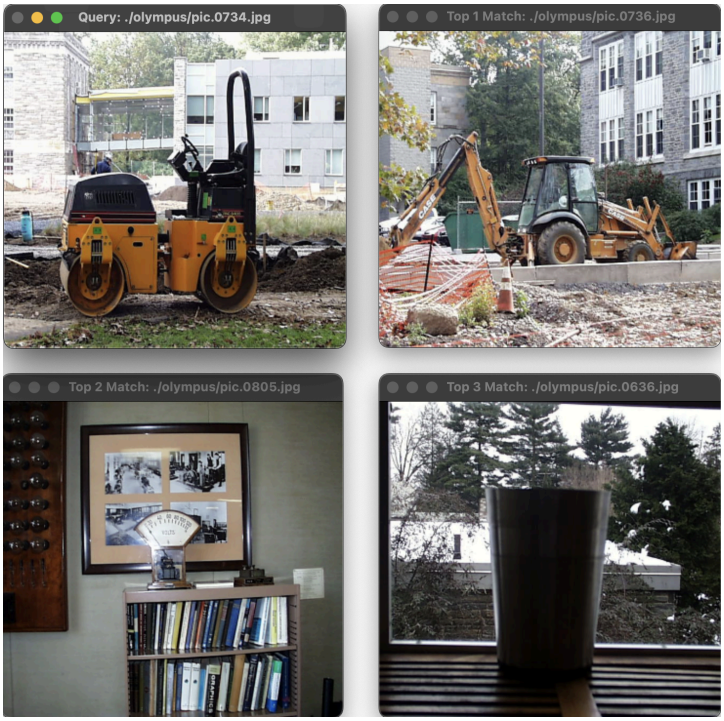
- Using Muti-Histograms:



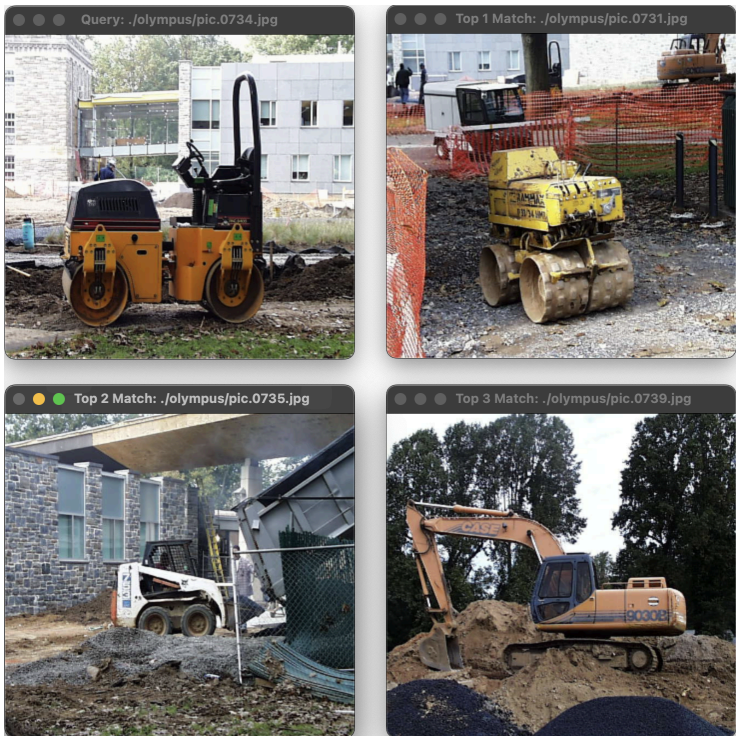
- Using deep neural networks:



- Top 3 matches for images *pic.0734.jpg:*
    - Using Multi-histograms:



    - Using deep neural networks:



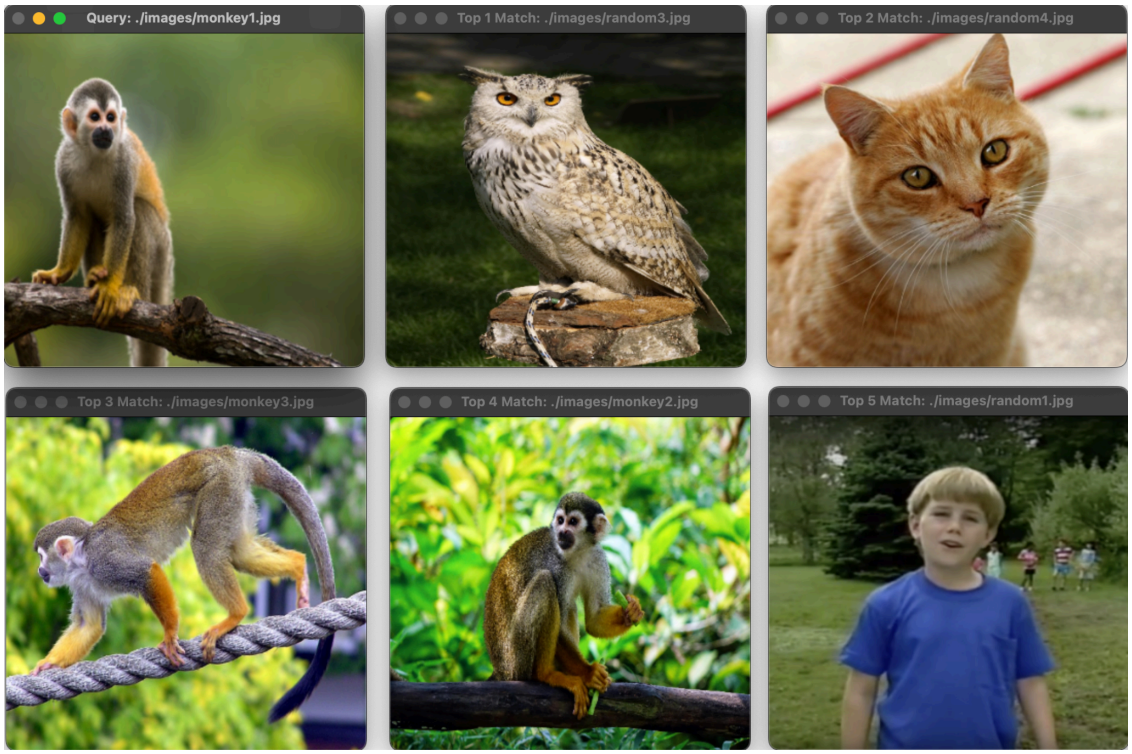
- We can see that every time, the deep neural network performs better than the multi-histogram technique every time. When the main focus is to identify objects, deep neural networks are always better. When we want to identify objects, the surroundings in the image become noise. Multi histograms do not understand this and capture everything but the deep neural networks learn to identify the object perfectly. In case 1, the white device is identified perfectly by DNN, where multi histograms focus on white objects with green background. In the second case, multi histograms fail completely. But in the last case, multi histogram captured a good top 1 match but failed at the rest. DNN prevailed in all these cases

7. **Custom design - Depth thresholding and RGB histograms (Req. image 7):** In this, instead of directly using RGB histograms, we first obtain depth values. We then apply a threshold to only use those pixels that are in the foreground for building an RGB histogram. This process takes more time compared to others but works to a certain extent. The metric used here is the histogram intersection. Below are the top 5 results obtained for image *pic.1074.jpg.*

- We can see that if the object is detected in the foreground perfectly, then we would get good results. If not, the results might not be perfect.

- I have also used a small database of 20 images that I collected to see matches for selected images. Below is the output that I got:

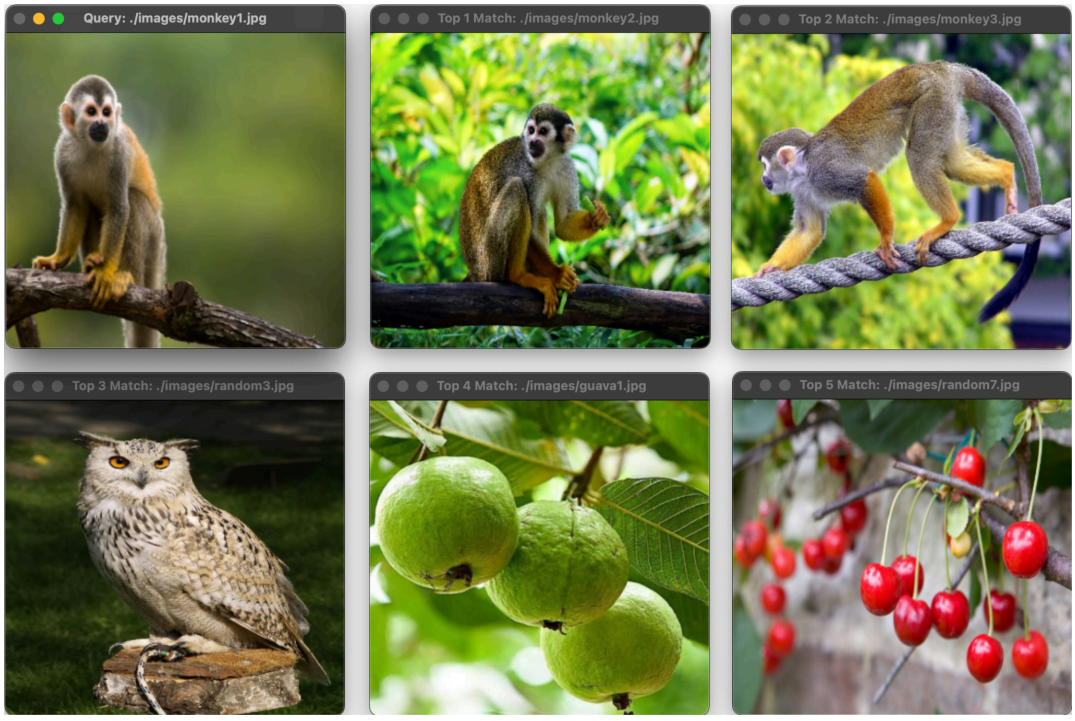
- Top 5 for query image:



- Worst matches for query image:

Extensions

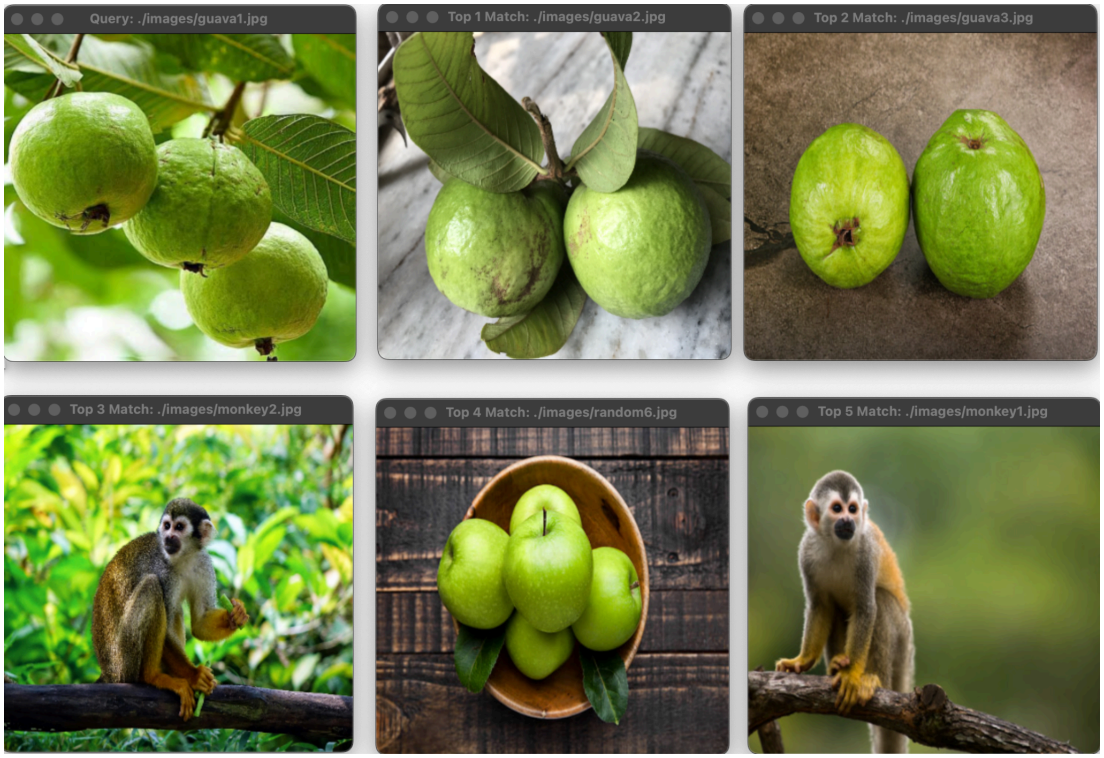
1. **Generating DNN embeddings:** Using the given sample code, I have incorporated the code to extract DNN embeddings into my project. I have used my small image database for finding matches and choosing a query with cosine distance as my evaluation metric. Below are the top 5 matches for the selected query.



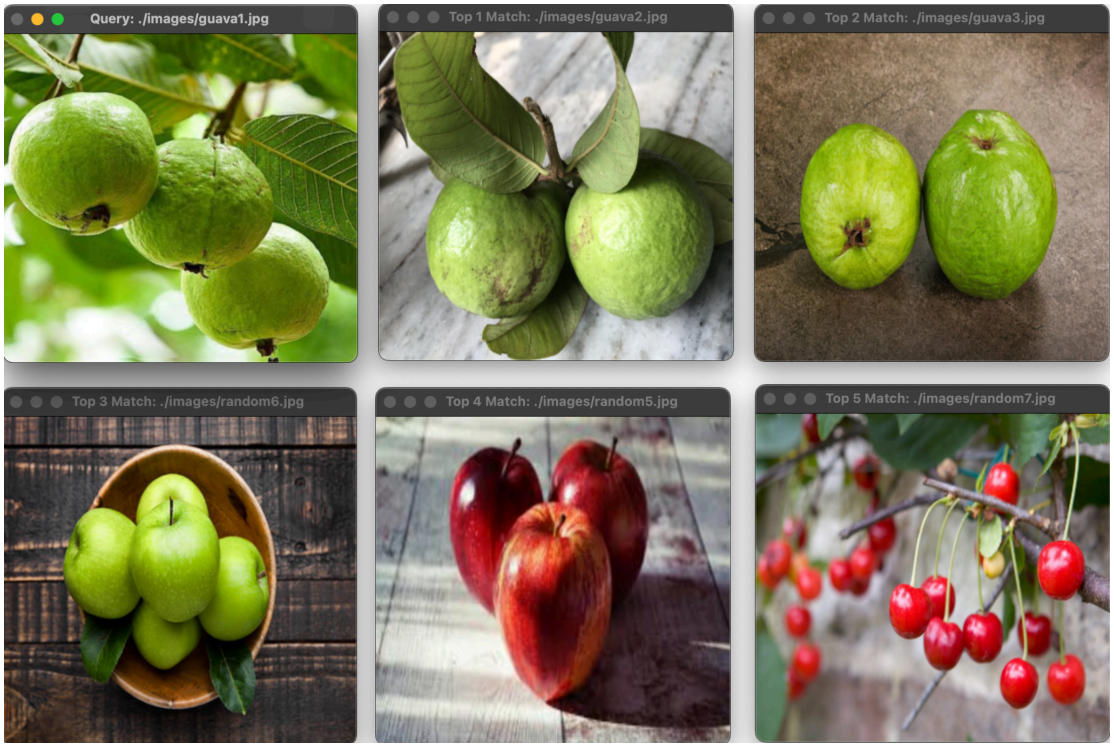
- We can see that DNN embeddings are still better.

2. **DNN embeddings with depth thresholding:** Instead of directly using DNN embeddings, I tried to blacken the background pixels and only use the foreground information to develop DNN embeddings. Cosine similarity is still the metric of evaluation.

- Depth DNN results:

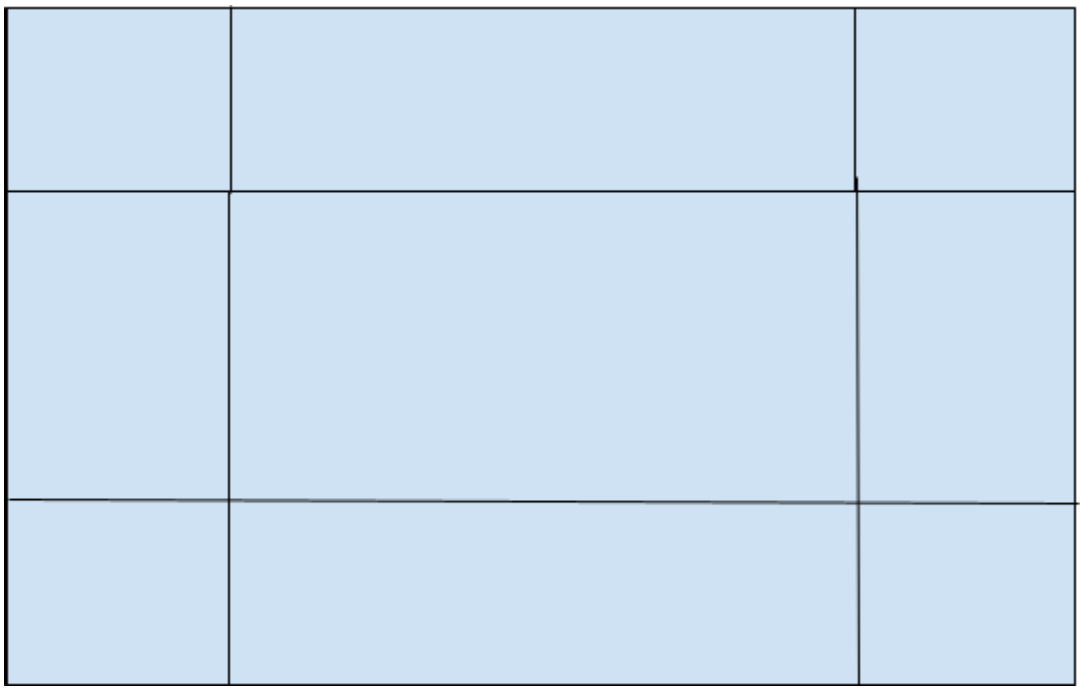


- Normal DNN results:

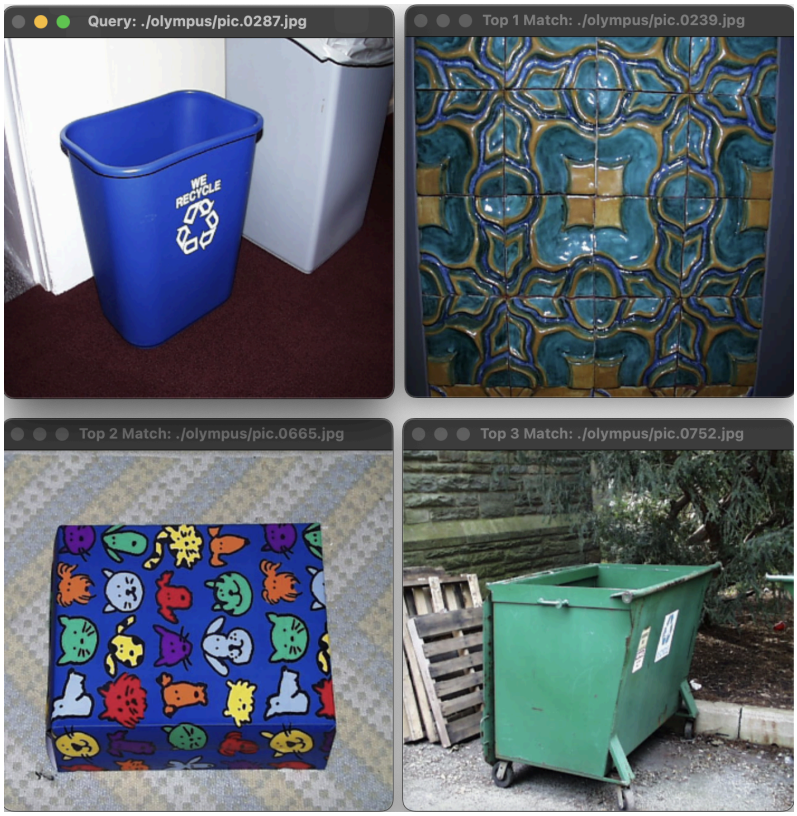


- We can still consider simple DNN to be more accurate as it is identifying all the fruits as top matches but depth DNN is not.

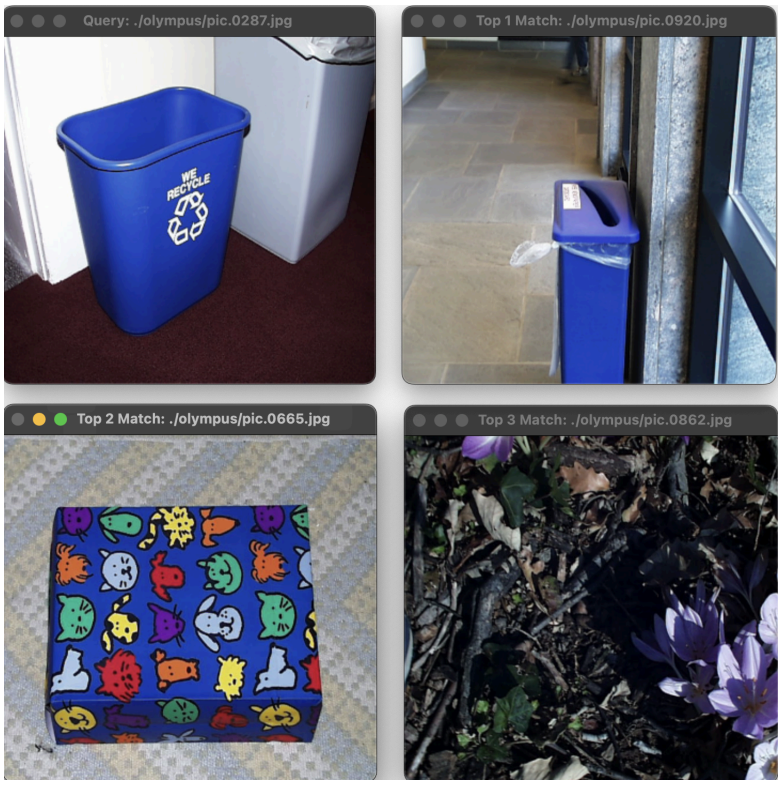
3. **4 intersecting histograms:** Instead of just two histograms, I have tried using 4 different RGB histograms for image as demonstrated below (middle rectangle is the intersection):



- Using these 4 histograms, I have computed an equally weighted Earth mover's distance metric (measures the minimum cost of transforming one distribution into another).
- Below are the top 3 results matches for *pic.0287.jpg.*



- Using cosine distance instead, we get the below results:



- We can see that cosine distance yielded better results compared to earth mover's distance. This indicates that not only choosing a good content based image retrieval technique is important, but it is also important to choose the correct evaluation metric.

Reflection

I had prior knowledge of Neural networks and how Convolutional neural networks work. But, I did not know any other content based image retrieval techniques. This project helped me to learn how content based image retrieval works using histograms, textures and DNN embeddings. It also helped me to understand the strengths and weaknesses of different techniques and also different evaluation metrics and the importance of choosing the right ones. Incorporating depth heavily relies on the separating background perfectly. If not, the results would be diluted.I hope to continue exploring content based image retrieval further to further improve my understanding of the concept.

Acknowledgements

The best resources I found for this assignment were stackoverflow and ChatGPT. I used stackoverflow and ChatGPT to understand errors that I encountered while working on this project. ChatGPT also helped me understand how few OpenCV functions work internally, the importance and theoretical knowledge about different techniques and evaluation metrics.The clear direction given in the project requirements made it easier to understand and implement all the tasks and compare my results to that of the given results.