Zhimin Liang

Spring 2024 CS5330

Project 2

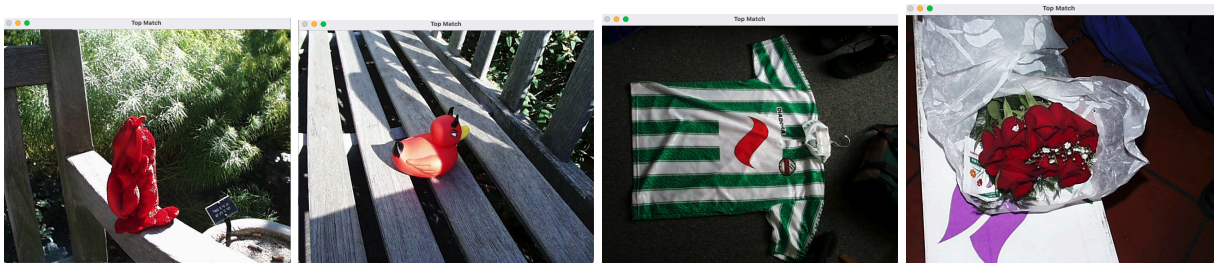
1.A short description of the overall project in your own words. (200 words or less)

The project is about to implement a Content-Based Image Retrieval (CBIR) system, and it incorporates both classic and modern approaches to feature extraction and comparison. Classic features include color histograms and texture-based histograms, while modern features involve using deep neural network (DNN) embeddings.The project allows us to specify a target image, and show the top match images in the database. The comparison metrics vary depending on the task, such as Sum of Squared Differences (SSD), and Histogram Intersection. Furthermore, the project includes customized tasks, such as comparing DNN embeddings against classic features for specific images and designing a unique CBIR system. In conclusion, The project aims to explore different feature representations and distance metrics in content-based image retrieval scenarios.

2.Any required images along with a short description of the meaning of the image.

Required result 1: *show the top three matches for the target image pic.1016.jpg*

```
lzm@xiaozhuizideMBP build % ./main ../olympus
Top 4 matches:
Image: ../olympus/pic.1016.jpg, Distance: 0
Image: ../olympus/pic.0986.jpg, Distance: 14049
Image: ../olympus/pic.0641.jpg, Distance: 21756
Image: ../olympus/pic.0547.jpg, Distance: 49703
```



This is the target image , and the top three matches, which are the same with requirment, We can find that the most distribution color in the middle 7 * 7 image is red

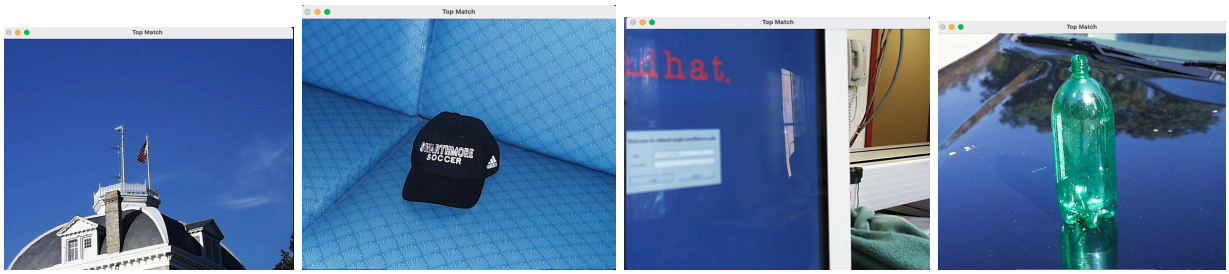
Function Implement:

-Implement the “computeFeatures” function to compute the features for a given image

-Implement the “computeSSDDistance” function to compute the sum-of-squared-difference distance between two feature vectors.

Required results 2: *show the top three matches for the target image pic.0164.jpg.*

```
lzm@xiaozhuizideMBP build % ./main ../olympus
Top 4 matches:
Image: ../olympus/pic.0164.jpg, Distance: 0
Image: ../olympus/pic.0599.jpg, Distance: 0.135059
Image: ../olympus/pic.0080.jpg, Distance: 0.307974
Image: ../olympus/pic.0898.jpg, Distance: 0.394244
lzm@xiaozhuizideMBP build %
```



pic.0164.jpg **pic.0599.jpg** **pic.0080.jpg** **pic.0898.jpg**

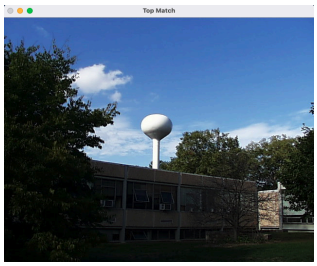
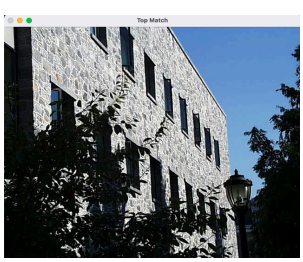
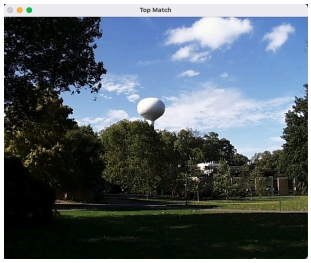
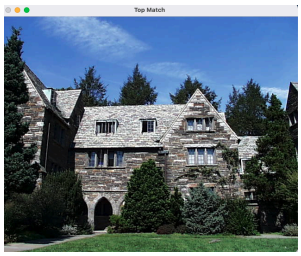
I use a whole image rg chromaticity histogram using 8 bins for each of r and g, and histogram intersection as the distance metric.We can find that the most distribution of color in the image is blue which is a dominance. And also with little white, gray and red, which corresponding to the target picture has the building roof.

Function Implement:

-Implement computeHistogram to calculate a single normalized color histogram for an image.
-Implement computeHistogramIntersection as the distance metric.

Required results 3: *show the top three matches for the target image pic.0274.jpg.*

```
Top 4 matches:
Image: ../olympus/pic.0274.jpg, Distance: -2
Image: ../olympus/pic.1031.jpg, Distance: -1.08655
Image: ../olympus/pic.0273.jpg, Distance: -1.08249
Image: ../olympus/pic.0409.jpg, Distance: -1.05888
lzm@xiaozhuizideMBP build %
```

pic.0274.jpg **pic.1031.jpg** **pic.0273.jpg** **pic.0409.jpg**

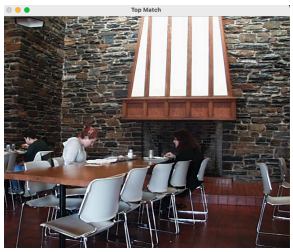
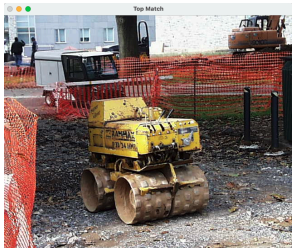
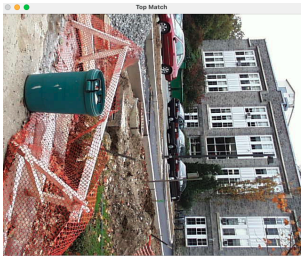
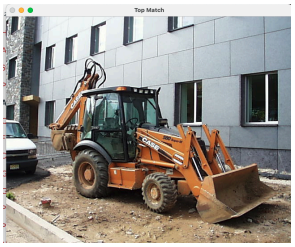
I use two RGB histograms, representing the top and bottom halves of the image, using 16 bins for each RGB and histogram intersection as the distance metric.The top half are mostly blue due to the sky, and the bottom half are mostly black and green since the shadow and trees.

Function Implement:

- The function divides the image into top and bottom halves.
- For each pixel, it extracts color channels (BGR) and calculates the bin index based on the specified number of bins.
- Histograms for the top and bottom halves are separately calculated and normalized by the total number of pixels.
- The two histograms are then combined into a single histogram vector.

***Required results 4**: show the top three matches for the target image pic.0535.jpg and show how they differ when compared to tasks 2 and 3.*

```
Top 4 matches:
Image: ../olympus/pic.0535.jpg, Distance: 0
Image: ../olympus/pic.0731.jpg, Distance: 0.00521688
Image: ../olympus/pic.1104.jpg, Distance: 0.00722594
Image: ../olympus/pic.0738.jpg, Distance: 0.0104389
```

pic.0535.jpg **pic.0731.jpg** **pic.1104.jpg** **pic.0738.jpg**

I use sum-of-squared-difference distance for both histograms, and take the average distance as distance metric.

About the color histogram: These results for query pic.0535 were obtained with a whole image rg chromaticity histogram using **16 bins** for each of r and g, so we can find that the pictures most have red , gray color.

About the Texture: texture histogram which calculates the Sobel magnitude image and uses a histogram of gradient magnitudes as texture feature.

These pictures have a regular texture with repetitive and predictable patterns, like the grids and the window, and they also have coarse textures, like rocks in the top three images, and surfaces with visible irregularities, because the target picture has uneven patterns on the wall.

If I use *pic.0535.jpg as target image for task 2*

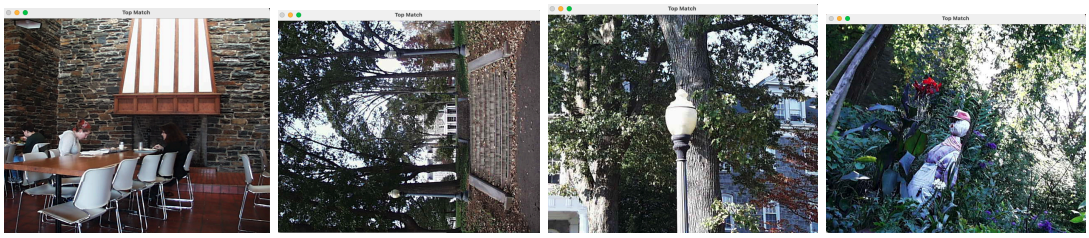
```
Top 4 matches:
Image: ../olympus/pic.0535.jpg, Distance: 1.19209e-07
Image: ../olympus/pic.0733.jpg, Distance: 0.12153
Image: ../olympus/pic.0731.jpg, Distance: 0.134427
Image: ../olympus/pic.0340.jpg, Distance: 0.137463
```

pic.0535.jpg pic.0733.jpg pic.0731.jpg pic.0340.jpg

I use a whole image rg chromaticity histogram using 16 bins for each of r and g and histogram intersection as the distance metric. We can find that the most distribution of color in the image is red and gray. And also with some white, which corresponds to the target picture that has a big gray wall with white fireplace, and the floor and desk is mostly red.

If I use *pic.0535.jpg as target image for task 3,*

```
lzm@xiaozhuizideMBP build % ./main ../olympus
Top 4 matches:
Image: ../olympus/pic.0535.jpg, Distance: -2
Image: ../olympus/pic.0952.jpg, Distance: -1.5532
Image: ../olympus/pic.0975.jpg, Distance: -1.54024
Image: ../olympus/pic.0431.jpg, Distance: -1.51535
```



pic.0535.jpg pic.0952.jpg pic.0975.jpg pic.0431.jpg

I use two RGB histograms, representing the top and bottom halves of the image, using 16 bins for each RGB and histogram intersection as the distance metric.The top half and the bottom half are mostly with gray and white, because the target picture have a background with a big gray wall and white fireplace on the wall.

Required results 5: *include the top 3 results for images pic.0893.jpg and pic.0164.jpg and compare the results with the prior methods.*

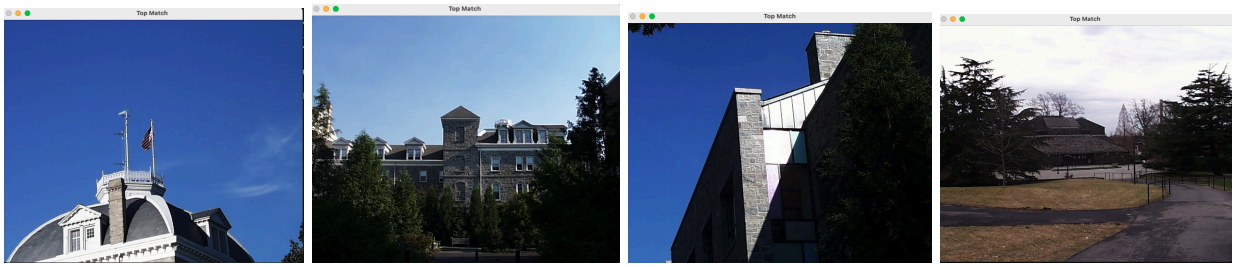
My top three matches of pic.0893.jpg :

```
Top 4 matches:
Image: ../olympus/pic.0893.jpg, Distance: 0
Image: ../olympus/pic.0897.jpg, Distance: 168.773
Image: ../olympus/pic.0136.jpg, Distance: 208.361
Image: ../olympus/pic.0885.jpg, Distance: 250.273
```

pic.0893.jpg **pic.0897.jpg** **pic.0136.jpg** **pic.0885.jpg**

My top three matches of pic.*0164*.jpg :

```
Top 4 matches:
Image: ../olympus/pic.0164.jpg, Distance: 0
Image: ../olympus/pic.0213.jpg, Distance: 149.049
Image: ../olympus/pic.1032.jpg, Distance: 172.578
Image: ../olympus/pic.0543.jpg, Distance: 179.482
```



pic.*0164*.jpg **pic.0213.jpg** **pic.1032.jpg** **pic.0543.jpg**

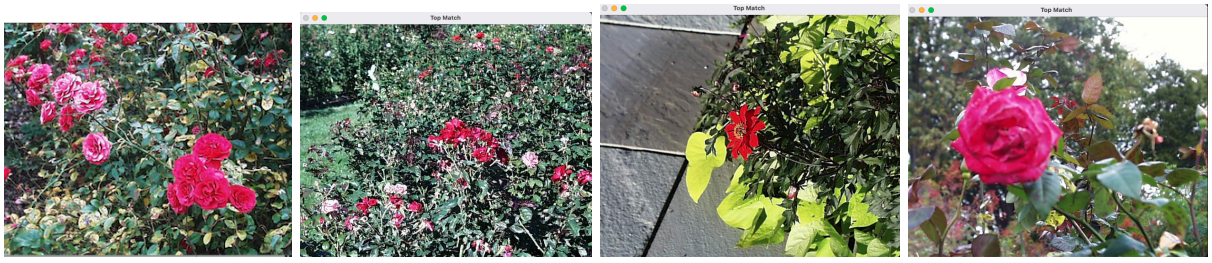
Compare the results with the prior methods:

For pic.*0164*.jpg in task 2, the result give us mostly base on the domain color in the histogram, but for DNN embedding, we can find that the result mostly give us a better matching, which means it will give us similar particular object, like the building or the water-pump.

Required result 6: *compare and contrast the DNN embedding and classic features results for 2-3 images of your choice.*

My top three matches of pic.1072.jpg using DNN embedding

```
Top 4 matches:
Image: ../olympus/pic.1072.jpg, Distance: 0
Image: ../olympus/pic.0143.jpg, Distance: 207.845
Image: ../olympus/pic.0329.jpg, Distance: 271.708
Image: ../olympus/pic.0940.jpg, Distance: 280.942
```

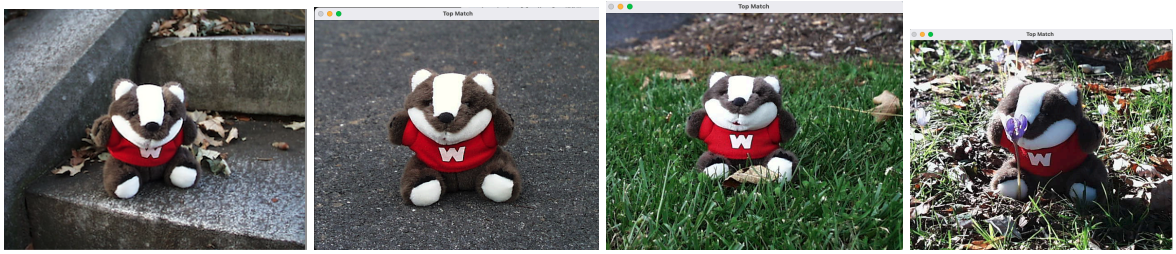


pic.1072.jpg **pic.0143.jpg** **pic.0329.jpg** **pic.0940.jpg**

This result shows that the DNN embedding gives us a highly similar matching for a particular object, all the top three images are flowers.

DNN embedding results of pic.0948.jpg

```
Top 4 matches:
Image: ../olympus/pic.0948.jpg, Distance: 0
Image: ../olympus/pic.0930.jpg, Distance: 183.172
Image: ../olympus/pic.0928.jpg, Distance: 281.205
Image: ../olympus/pic.0960.jpg, Distance: 282.019
```



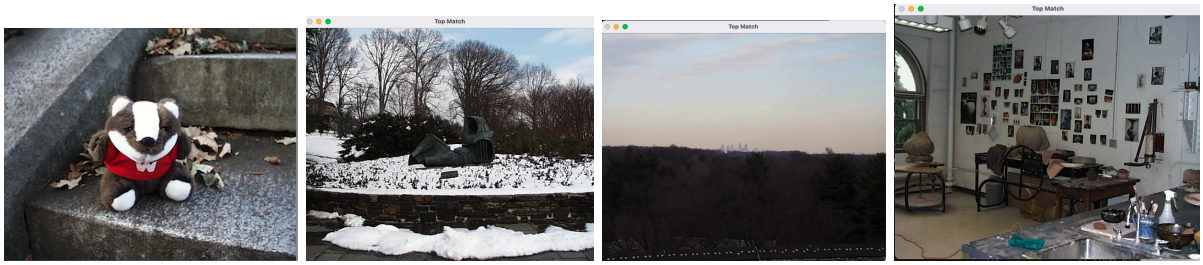
pic.0948.jpg **pic.0930.jpg** **pic.0928.jpg** **pic.0960.jpg**

This result shows that the DNN embedding gives us a highly similar matching for a particular object, all the top three images are the same toy.

Classic features results of pic.0948.jpg

```
Top 4 matches:
Image: ../olympus/pic.0948.jpg, Distance: 0
Image: ../olympus/pic.0629.jpg, Distance: 0.0448944
Image: ../olympus/pic.0610.jpg, Distance: 0.0462891
Image: ../olympus/pic.0007.jpg, Distance: 0.0526062
```

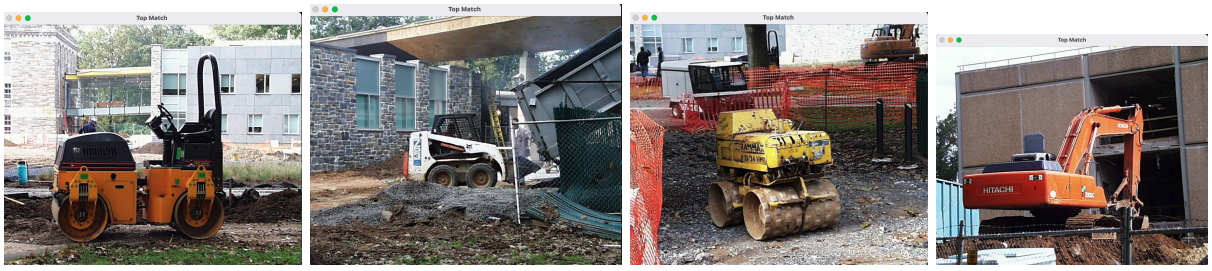
metric.



However, when I use Classic features rg chromaticity histogram with 8 bins and histogram intersection as the distance, the results shows that they match the majority color with the target, which is gray and white on the background and toy, other than tol find the particular object ad the DNN.

DNN embedding results of pic.0734.jpg

```
Top 4 matches:
Image: ../olympus/pic.0734.jpg, Distance: 0
Image: ../olympus/pic.0735.jpg, Distance: 179.449
Image: ../olympus/pic.0731.jpg, Distance: 187.584
Image: ../olympus/pic.0745.jpg, Distance: 190.18
```

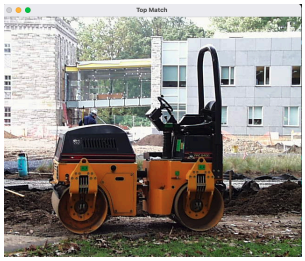
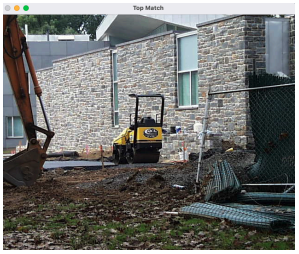
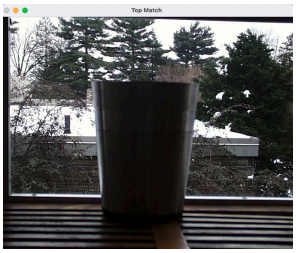


pic.0734.jpg **pic.0735.jpg** **pic.0731.jpg** **pic.0745.jpg**

Here shows the same as previous images using DNN embedding, which give us the similar object.

Classic features results of pic.0948.jpg

```
Top 4 matches:
Image: ../olympus/pic.0734.jpg, Distance: 0
Image: ../olympus/pic.0741.jpg, Distance: 0.00205085
Image: ../olympus/pic.0636.jpg, Distance: 0.0021537
Image: ../olympus/pic.0736.jpg, Distance: 0.00291611
```


pic.0734.jpg **pic.0741.jpg** **pic.0636.jpg** **pic.0736.jpg**

These results are captured using a whole image color histogram and a whole image texture histogram as the feature vector and sum-of-squared-difference distance as distance. We can find that it shows a better match than just using color histogram, but it is still weaker than DNN embarding which gives us all the car objects.

Conclusion:

Base on the above comparison for pic.0948.jpg and pic.0734.jpg

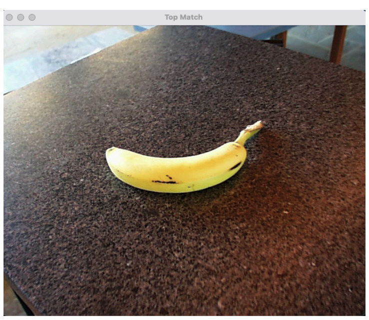
I think DNN embeddings will always perform better than the classical features, because DNN embeddings are learned through the training process of neural networks. Therefore, the network adjusts its weights to minimize a specified objective which leads to the learning of meaningful representations in the lower-dimensional embedding space.

Required results 7: *for two target images of your choice, show the top five results. It's also helpful to show some of the least similar results.*

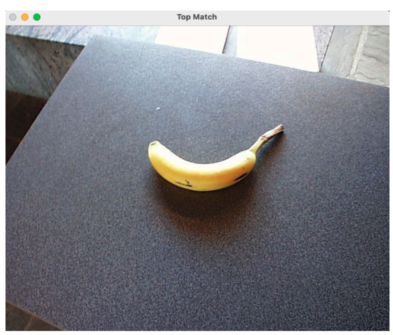
Design: I combine texture histogram and DNN embeddings into a unified feature vector, and then use cosine distance as distance metric.

Top five match of target picture pic.0343.jpg

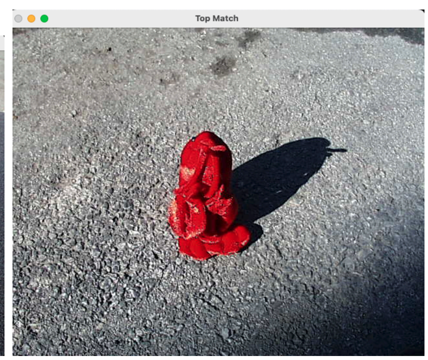
```
Top 6 matches:
Image: ../olympus/pic.0343.jpg, Distance: 0
Image: ../olympus/pic.0345.jpg, Distance: 0.11332
Image: ../olympus/pic.1010.jpg, Distance: 0.267972
Image: ../olympus/pic.0496.jpg, Distance: 0.273459
Image: ../olympus/pic.0471.jpg, Distance: 0.275711
Image: ../olympus/pic.0346.jpg, Distance: 0.280848
```



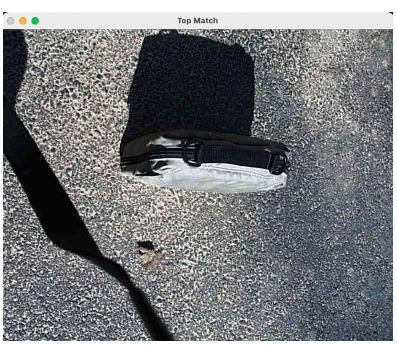
pic.0343.jpg



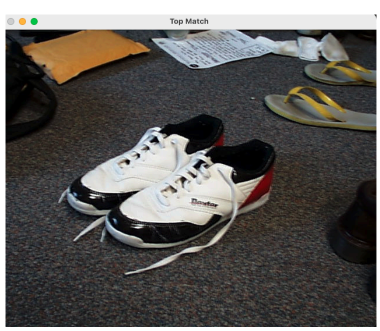
pic.0343.jpg



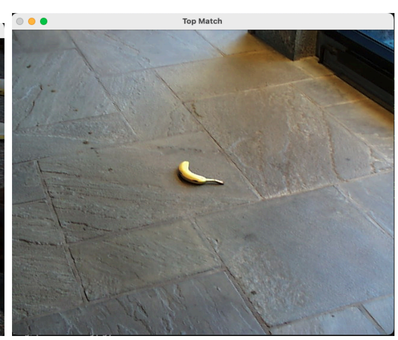
pic.0343.jpg



pic.0343.jpg



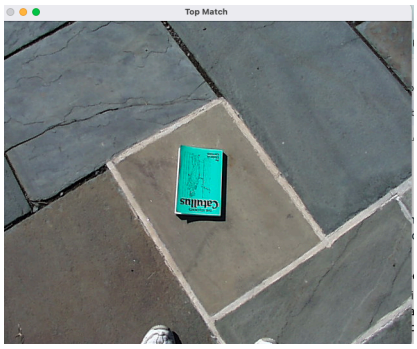
pic.0343.jpg



pic.0343.jpg

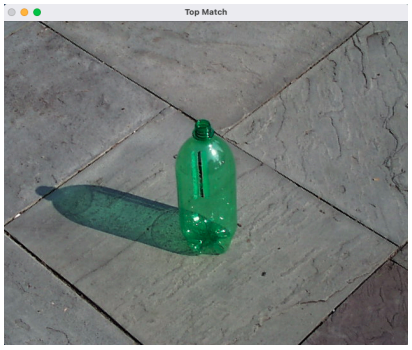
Top five match of target picture pic.0375.jpg

```
Top 6 matches:
Image: ../olympus/pic.0375.jpg, Distance: -1.19209e-07
Image: ../olympus/pic.0118.jpg, Distance: 0.200414
Image: ../olympus/pic.0908.jpg, Distance: 0.226581
Image: ../olympus/pic.0559.jpg, Distance: 0.235589
Image: ../olympus/pic.1046.jpg, Distance: 0.23985
Image: ../olympus/pic.0116.jpg, Distance: 0.242294
```



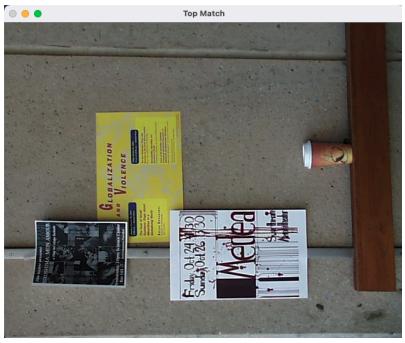
pic.0375.jpg

pic.0118.jpg

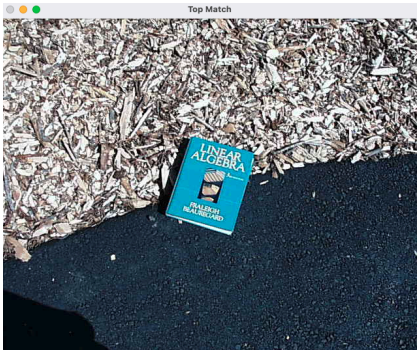


pic.0908.jpg

pic.0559.jpg



pic.1046.jpg



pic.0116.jpg

The texture histogram captures fine details and patterns within the image, assigning significance to the spatial distribution of pixel intensities. On the other hand, DNN embeddings contribute higher-level semantic information, capturing complex patterns and features that might not be apparent in the texture histogram.

The choice of using Cosine distance as the distance metric emphasizes the alignment or orientation of these feature vectors rather than their magnitudes.

Combining this information, it seems that the texture information tends to weigh more in the matching process, that might be because the target I choose is not very complex, but DNN embeddings are useful when capturing complex patterns.

3. A description and example images of any extensions.

None

4.A short reflection of what you learned.

This project improves my understanding of content-based image retrieval and the importance of feature selection and distance metrics in achieving meaningful results. The diverse tasks allowed me to explore different approaches and understand their strengths and limitations, which give me experience in extracting various image features, including color histograms, texture histograms, and deep network embeddings. And I also Understood the significance of choosing appropriate distance metrics based on the nature of features, impacting the effectiveness of the CBIR system.In addition, this project improved my C++ programming skills and gained practical insights into image processing techniques.

5.Acknowledgement of any materials or people you consulted for the assignment.

TAs: Tejaswini Dilip Deore

OpenCV Tutorials: https://docs.opencv.org/4.5.1/d9/df8/tutorial_root.html

YouTube: Computer Vision Lab, “OPENCV & C++ TUTORIAL”,https://www.youtube.com/playlist?list=PLUTbi0GOQwghR9db9p6yHqwvzc989q_mu [access, Feb,14,2024]