Report 2 – CBIR: Content Based Image Retrieval

@Yanting Lai

Description

This project involves building a Content-based Image Retrieval (CBIR) system to find similar images in a database. It starts with baseline matching using a small image region and sum of squared differences, followed by color histogram matching. It then incorporates multi-histogram and texture features, using Sobel filters and color histograms. The project also applies deep network embeddings from ResNet18 for comparison, evaluating both classic features and embeddings. Finally, a custom feature vector and distance metric are developed for specific image types to further experiment with retrieval methods.

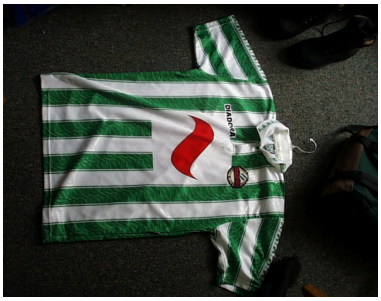
Task 1: Baseline Matching

> The objective of this task is to use a 9x9 central square from the image as the feature vector and compare it to other images by calculating the sum-of-squared-differences as the distance metric. The program should ensure that when comparing the image with itself, the distance is zero.

Target Image (pic.1016.jpg):

Top three matches:

pic.0986.jpg



pic.0641.jpg

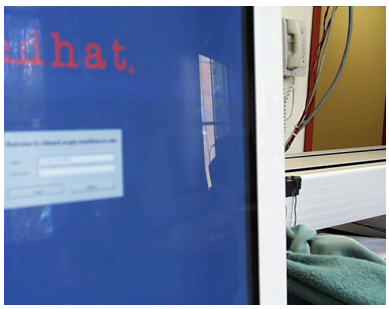
pic.0574.jpg

Task 2: Histogram Matching

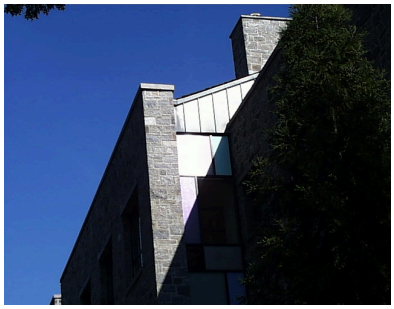
> In this task, we use a normalized color histogram as the feature vector. Specifically, we use an RGB histogram with 8 bins for each color channel. The histogram is then normalized by dividing by the total number of pixels in the image. To measure similarity between images, we apply histogram intersection as the distance metric.

Target Image (pic.0164.jpg):

Top three matches:



Pic.0080.jpg



pic.1032.jpg

pic.0898.jpg

Task 3: Multi-Histogram Matching

> In Task 3, I applied a multi-histogram matching using color histograms from different image regions. A custom distance metric, combining histogram intersection and weighted averaging, helps identify the top three matches for image pic.0274.jpg.

Target Image (pic.0274..jpg):

Top three matches:

Pic.0825.jpg

pic.0275.jpg

pic.1055.jpg

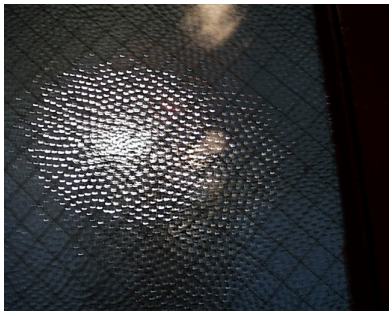
Task 4: Texture and Color

> This task involves using both color and texture histograms as the feature vector for image matching. The texture metric can be based on the Sobel magnitude, gradient orientation, or other advanced methods like co-occurrence matrices, Laws filters, Gabor filters, or Fourier transforms. The goal is to create a balanced distance metric that weighs both color and texture equally. The task also encourages experimenting with different texture analysis methods to observe their impact on matching results.

Target Image (pic.0535..jpg):

Top three matches with texture and color matching method:

Pic.0537.jpg



pic.0845.jpg

pic.0411.jpg

Top three matches with histogram matching method:

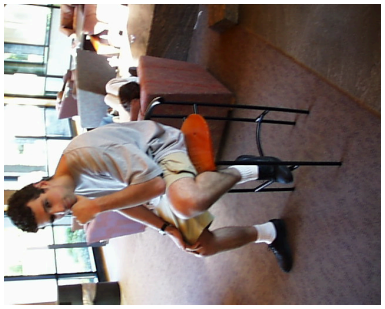
Pic.0573.jpg

pic.0217.jpg

pic.0172.jpg

Top three matches with multi- histogram matching method:

Pic.0089.jpg



pic.0339.jpg

pic.0340.jpg

Task 5: Deep Network Embeddings

> Task 5 uses precomputed ResNet18 embeddings from a CSV file to match images. The task involves selecting a distance metric, like sum-squared or cosine distance, to find the top 3 matches for 0893 and 0164. The results are compared to previous methods, highlighting differences in using deep network embeddings for image matching.

Deep Network Embeddings matching:


Pic.0893.jpg

pic.0223.jpg


pic.0979.jpg


pic.0032.jpg


Pic.0164.jpg


pic.1072.jpg

pic.0223.jpg

pic.0036.jpg

Texture and color matching:


Pic.0893.jpg


Pic.1007.jpg


pic.0781.jpg


pic.0485.jpg


Pic.0164.jpg


Pic.0433.jpg


pic.0047.jpg


pic.0556.jpg

Reflection: Why doesn't the matching results visually make sense?

1) **Abstract feature embeddings**:

ResNet18 extracts high-dimensional abstract features, focusing more on shapes, edges, and fine details rather than the semantic content of the image.

2) **Limitations of the matching algorithm**:

The distance metrics used (e.g., cosine distance, sum of squared differences) measure vector similarity mathematically, but this doesn't necessarily align with human visual perception.

3) **Model training bias:**

ResNet18 was trained on ImageNet, and it may not capture specific features of fire hydrants well, leading the model to overemphasize other image features.Thus, the matching results tend to favor images with visual detail similarities, even if the semantic content is unrelated.

Comparison between Deep Network Embeddings and Texture and Color Matching

Aspect	Deep Network Embeddings	Texture and Color Matching
Features	High-level abstract details from deep learning	Low-level details like color and texture
Focus	Shape and structure	Color distribution and surface patterns
Distance Metric	Sum-squared or cosine distance	Weighted color and texture distances
Strengths	Captures complex, deep patterns	Matches based on visual, surface-level similarities
Weaknesses	Can give unexpected or abstract matches	Limited to simpler visual features
Best For	Complex, deep feature-based tasks	Visual similarity based on color and texture

6.Compare DNN and other classic methods

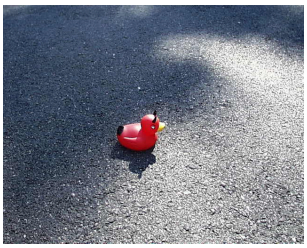
> This task aims to pick 2-3 images and compare the results of using DNN Embeddings versus classic features. The goal is to test the DNN embedding and other methods on various images and see what's the difference between these two methods.

Texture and Color method (1072, 0688, 0983, 0984)


Pic.1072.jpg


Pic.0688.jpg


pic.0983.jpg


pic.0984.jpg

DNN method (1072, 0032, 0033, 0979)


Pic.1072.jpg


Pic.0032.jpg


pic.0033.jpg


pic.0979.jpg

Histogram method (948, 0620, 0564, 0471)


Pic.0948.jpg

Pic.0620.jpg


pic.0564.jpg


pic.0471.jpg

DNN method (948, 1064 , 0223, 0949)


Pic.0948.jpg


Pic.1064.jpg

pic.0223.jpg


pic.0949.jpg

DNN method (0734, 0918 , 1064, 1068)


Pic.0743.jpg


Pic.0918.jpg


pic.1064.jpg


pic.1068.jpg

Multi-Hist method (0734, 0736, 0795, 0207)


Pic.0743.jpg


Pic.0736.jpg


pic.0795.jpg


pic.0207.jpg

In comparing DNN embeddings with classic methods, the key findings are:

DNN Embeddings often capture abstract textures and details, leading to diverse matches that might miss contextual relevance. For example, in image 1072, DNN matches unrelated objects like pens and pianos, while the classic method finds visually similar outdoor scenes.

Classic Methods (texture, color, histograms) tend to focus on direct visual and contextual similarities, making them more reliable for images where color and structure matter, like in image 948 (plush toy) and 734 (construction scene).

Overall, DNN is better for abstract features, while classic methods are stronger for visual context.

7.Custom CBIR

> This task involves creating a CBIR system using a custom feature vector and distance metric, combining DNN embeddings with other features. After developing the system with a small training set, you'll evaluate it by selecting two target images, showing the top five most similar results, and comparing them to expected outcomes.

Target image: pic.1011.jpg

Top 5 Most Similar Items


pic.0368.jpg


pic.0495.jpg

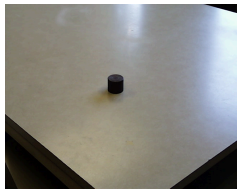

pic.0123.jpg

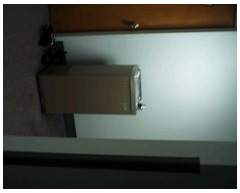

pic.0121.jpg


pic.0981.jpg

Top 5 Least Similar Items


pic.0511.jpg


pic.0400.jpg


pic.0443.jpg


pic.0034.jpg


pic.0598.jpg

Target image: pic.0358.jpg

Top 5 Most Similar Items


pic.1031.jpg

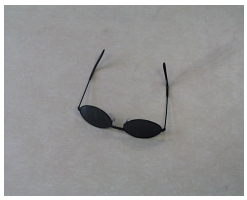

pic.1023.jpg


pic.0452.jpg


pic.0450.jpg


pic.1055.jpg

Top 5 Least Similar Items


pic.0512.jpg


pic.0225.jpg


pic.0035.jpg


pic.0558.jpg


pic.0311.jpg

8.Reflection

I had two difficulties when completing this assignment:

1. DNN embeddings don't seem to identify objects as I would expect

The outcome doesn't visually make sense since it could have very different objects in very different environment settings with different colors and textures. I've tried various methods and I think there are several possibilities to this issue:

-The target image that I chose

-The embedding parameters might not be the best representation for the 2D picture

2.Custom CBIR design is far from perfect

Initially, I was planning to choose a Kaggle data set to create an object classification that can tell cats from dogs. However, there are several technical details that delayed the submission of this project:

-I need to generate a custom embedding file for the dataset

-The dataset is very large so I need to write a bash script to automatically run it

-One the embedding file was generated, I realized the embedding vectors are not aligning with the picture name

-The combined method used weighted average of different methods to calculate this custom CBIR feature and maybe this is not the best way

-The target image I choose would also change the outcome and it is unstable

3.Some extra thoughts on this assignment

After this assignment, I've had some more new questions:

What is a good way to make DNN identify objects in a better way? And does the reCAPTCHA have anything to do with this current project that we are doing? Is identifying objects a difficult task, a simple thing for humans to do but difficult for a computer? And what about the apple face ID, how does it make sure the identification can be accurate under different light circumstances?

9.Learning and links

1. **OpenCV Documentation**
   https://docs.opencv.org/
2. **Image Processing Techniques in Computer Vision**
   https://www.tutorialspoint.com/dip/index.htm
3. **C++ Programming Language Reference**
   https://en.cppreference.com/w/
4. **Deep Learning with ResNet**
   https://arxiv.org/abs/1512.03385
5. **Content-Based Image Retrieval (CBIR)**
   https://en.wikipedia.org/wiki/Content-based_image_retrieval
6. **Histogram of Oriented Gradients (HOG) for Image Recognition**
   https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf
7. **Kaggle Datasets for Image Retrieval**
   https://www.kaggle.com/datasets
8. **Edge Detection Techniques: Sobel, Canny**
   https://en.wikipedia.org/wiki/Canny_edge_detector
9. **Deep Learning for Image Classification: ResNet & CNNs**
   https://cs231n.github.io/convolutional-networks/