Project 2: Content-based Image Retrieval

Parker Cai
NUID: 002529785
02/07/2026

Jenny Nguyen
NUID: 002677046
02/07/2026

Project description:

This project builds an image search system that finds similar images based on what they look like. Instead of searching by tags or filenames, the system actually analyzes the visual content of images and their colors, textures, and patterns.

We built several different methods to compare images. The simplest one just looks at the center pixels. Others use color histograms, which track how much of each color appears in an image, split the image into regions to capture layout, or analyze texture patterns using edge detection. In addition to these classical image features, we also tested using pre-trained neural network embeddings that can recognize higher-level features. We also designed a custom method that combines DNN embeddings with skin tone and brightness detection for better portrait matching. To speed up experimenting with all these methods, we built an interactive GUI using Dear ImGui that lets us visually compare results side by side.

Task Results:

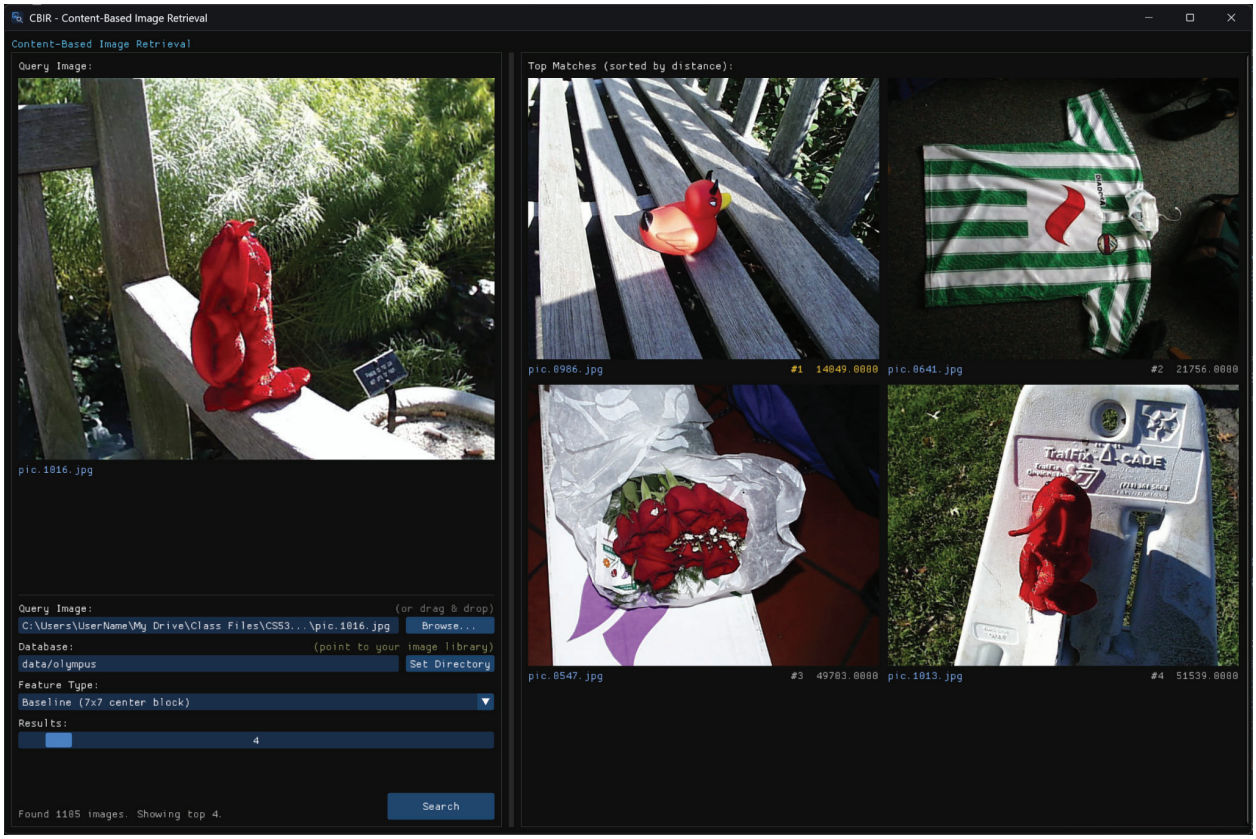
Task 1: Baseline Matching

Takes the 7×7 square in the center of the image and compares pixels directly using sum-of-squared-difference(SSD).

Query Image: pic.1016.jpg

Top 3 Matches:

- pic.0986.jpg (distance: 14049.0)
- pic.0641.jpg (distance: 21756.0)
- pic.0547.jpg (distance: 49703.0)



Description: The baseline method matches images by comparing only a 7x7 pixel block in middle of the image, so the results all share a similar bright red color in their center region. This explains why a red duck, a jersey with a red logo, red roses, and another red lobster were returned as top matches despite being very different subjects.

Task 2: Histogram Matching

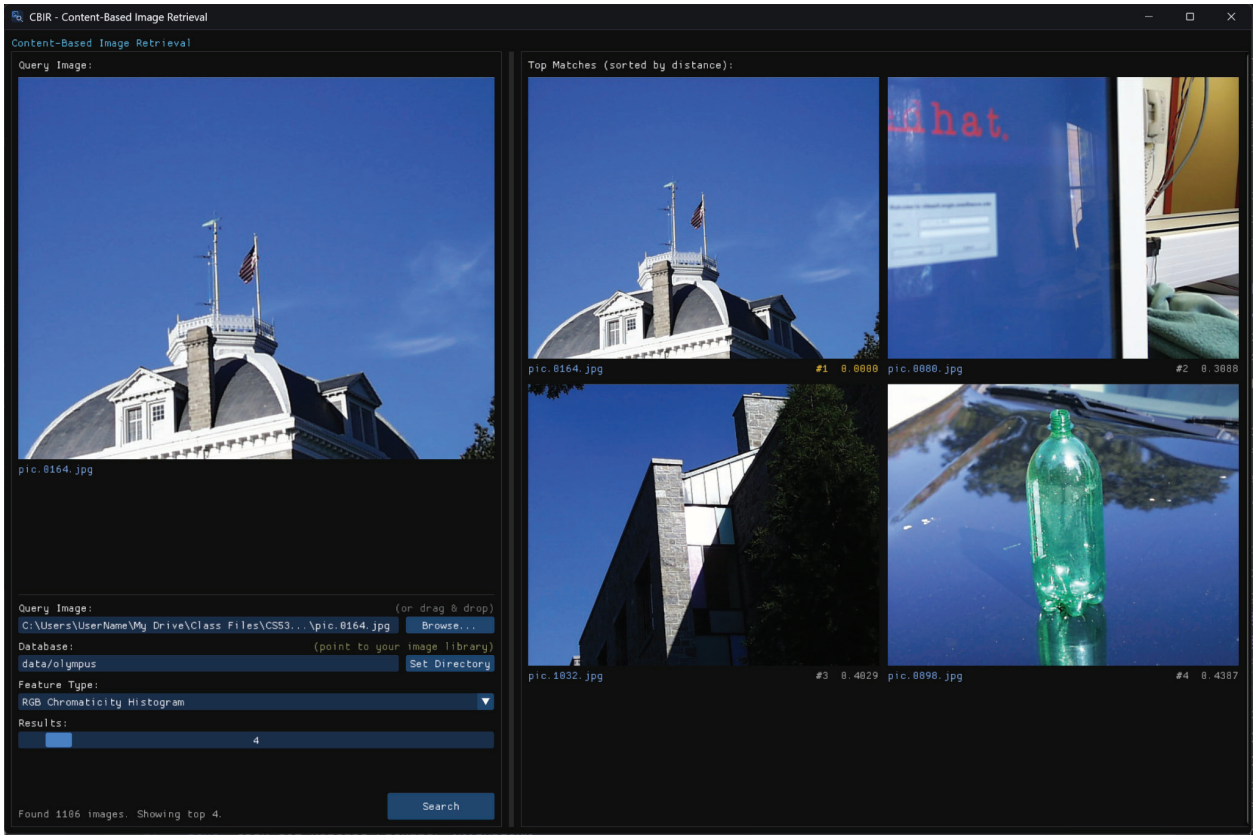
Makes a 2D histogram of RG chromaticity values that compares using histogram intersection.

Query Image: pic.0164.jpg

Top 3 Matches:

- pic.0080.jpg (distance: 0.308557)
- pic.0489.jpg (distance: 0.344745) - would be pic.1032.jpg w/ formula: (bin(x) = min(floor(x * bins), bins - 1)
- pic.0461.jpg (distance: 0.356031)

Description: The RG chromaticity histogram matches images that have a similar overall color distribution. This is why an indoor scene with blue walls, a squirrel on gray-blue pavement, and a blue-jeans background all matched the query image, which had mostly blue sky.



The RGB chromaticity histogram match images show very similar results.

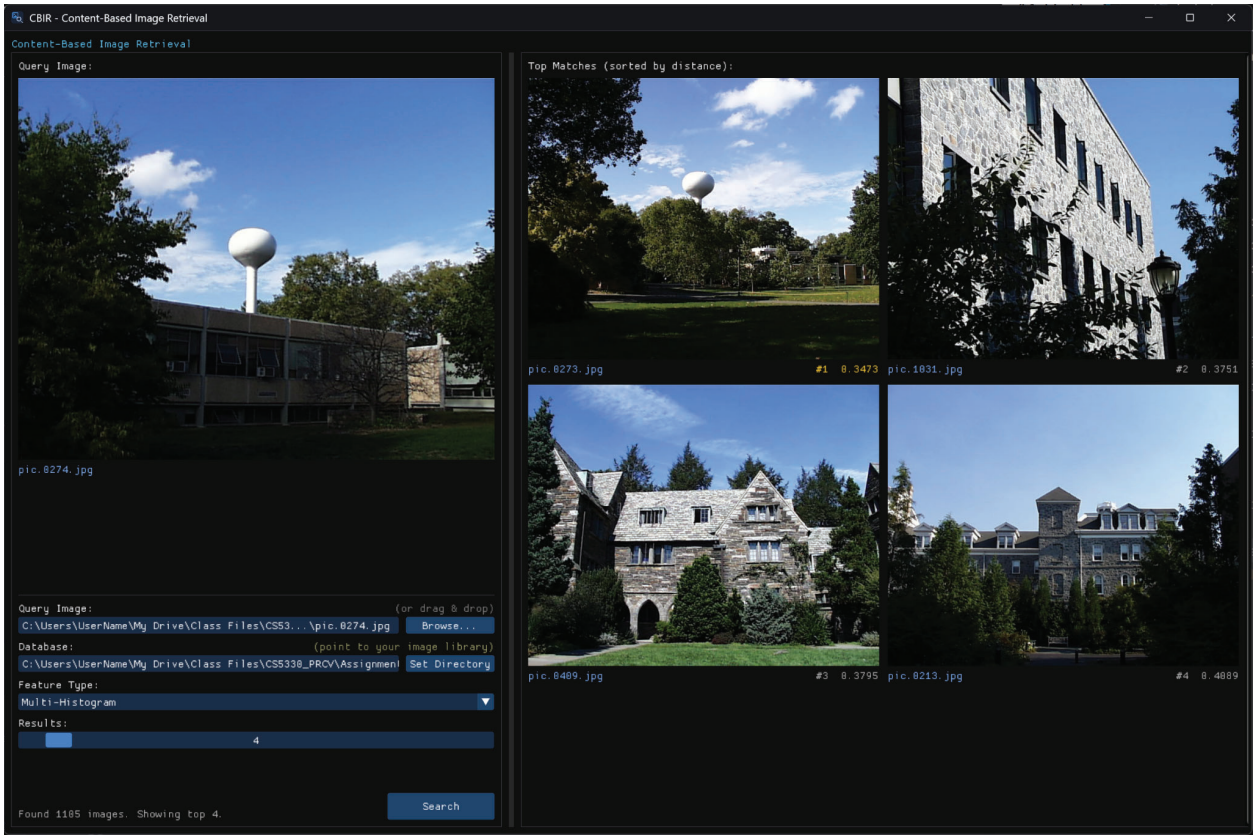
Task 3: Multi-histogram Matching

Splits the image into top and bottom halves, makes separate RGB histograms for each half, and compares with 50/50 weighting.

Query Image: pic.0274.jpg

Top 3 Matches:

- pic.0273.jpg (distance: 0.347339)
- pic.1031.jpg (distance: 0.375110)
- pic.0409.jpg (distance: 0.379523)

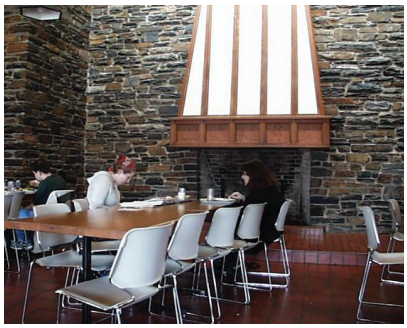


Description: This captures both color and rough layout by keeping top/bottom separate. The matches have similar sky colors on top and similar shadows of the trees on the bottom. This works well when it is an outdoor image with sky on top and ground on bottom.

Task 4: Texture and Color

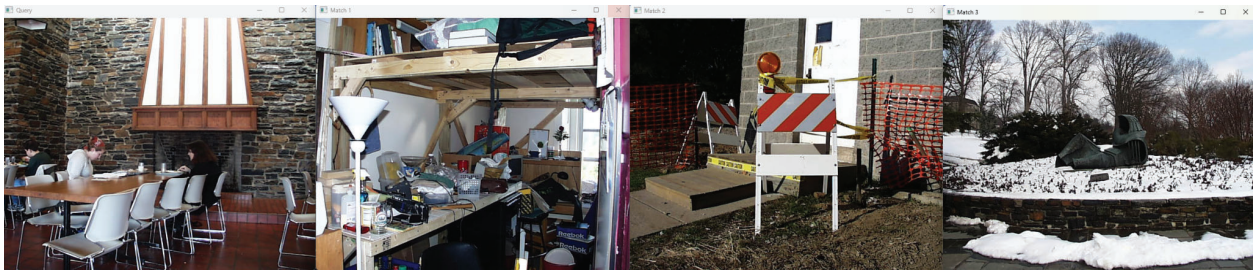
Combines Sobel edge magnitude histogram with RGB color histogram and equal weighting.

Query Image: pic.0535.jpg



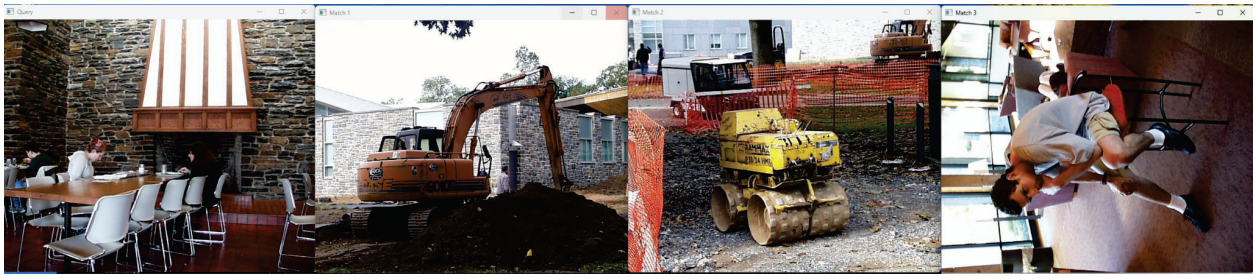
Top 3 Matches:

- pic.0171.jpg (distance: 0.185832)
- pic.0454.jpg (distance: 0.188620)
- pic.0629.jpg (distance: 0.190652)



Comparing with Task 2: Histogram only
Top 3 Matches:

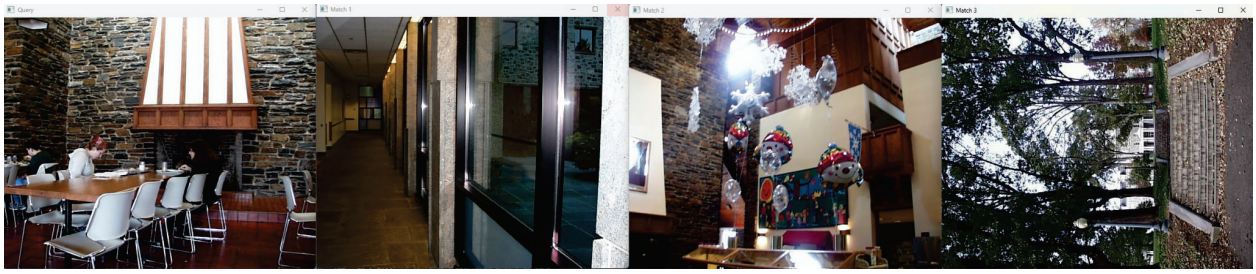
- pic.0733.jpg (distance: 0.121530)
- pic.0731.jpg (distance: 0.134427)
- pic.0340.jpg (distance: 0.137463)



Comparing with Task 3: Multi-histogram

Top 3 Matches:

- pic.0285.jpg (distance: 0.277176)
- pic.0628.jpg (distance: 0.326770)
- pic.0952.jpg (distance: 0.329404)



Description: We used the Sobel gradient magnitude histogram as the texture metric. The results are completely different depending on which method you use, which shows that feature choice really matters. Adding texture helps find images that are similar, not just the color tones.

In the first matching of just texture and color, it totally gave a different match. The cafeteria has lots of edges and structure where the first and second match have similar edge patterns. They all share both the colors and structural patterns.

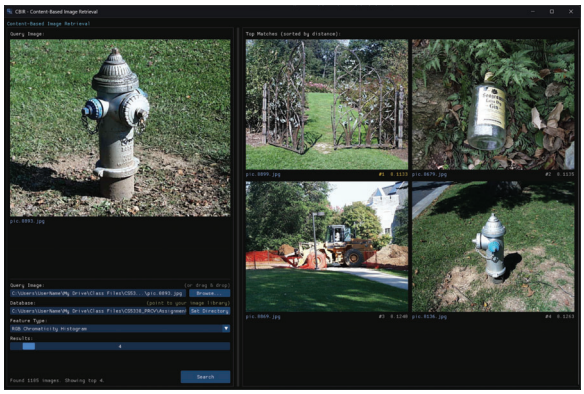
Now comparing task 2 with the histogram, they all have similar brown and tan colors, but they don't really look similar at all. They all just happen to have similar color tones.

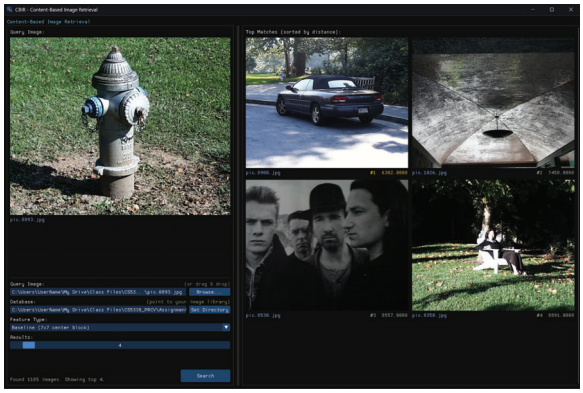
Now comparing task 3 with the multi-histogram, it at least has some spatial layouts like with darker colors on the bottom and lighter colors on top. It seems like it focused more on the color placement rather than what the surfaces actually looked like.

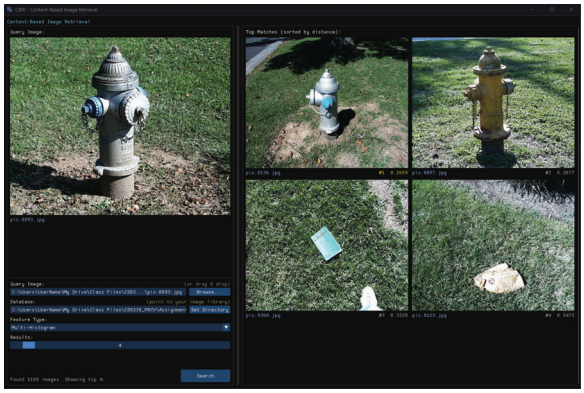
Task 5: Deep Network Embeddings

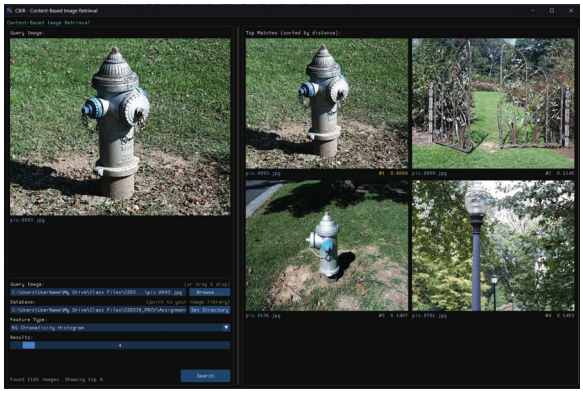
Required results 5: include the top 3 results for images pic.0893.jpg and pic.0164.jpg and compare the results with the prior methods.

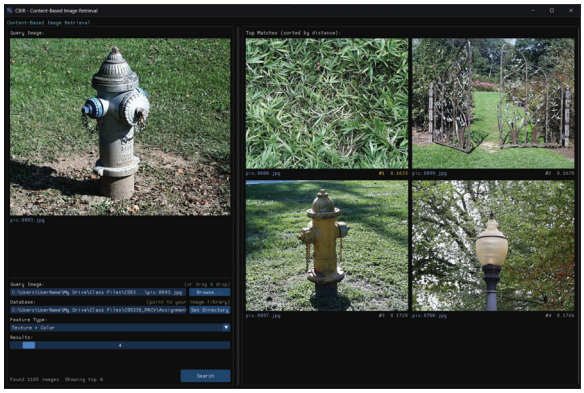
pic.0893.jpg





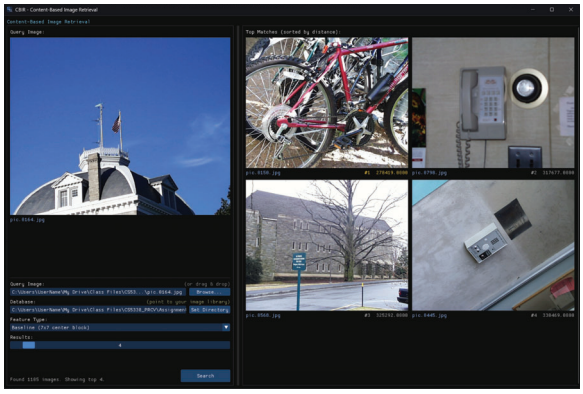


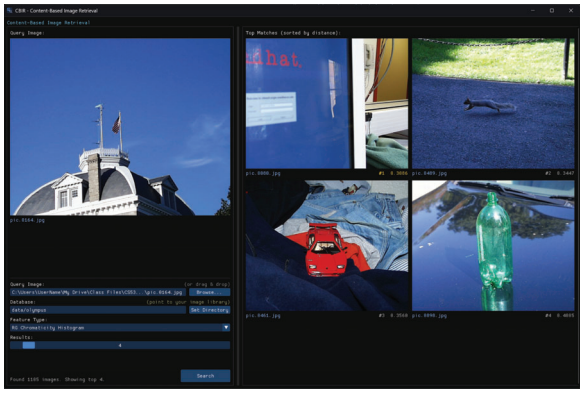




pic.0164.jpg







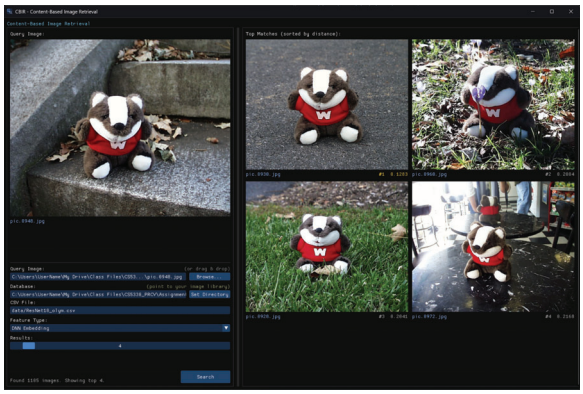
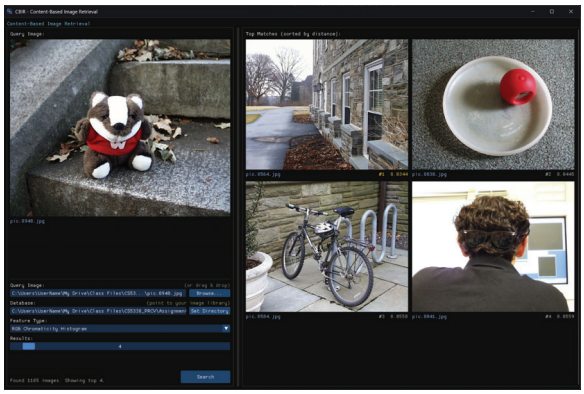
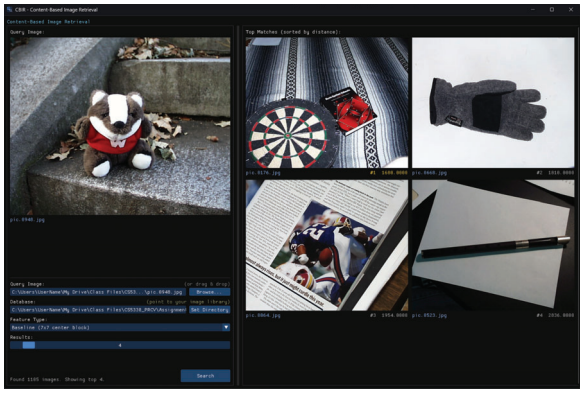
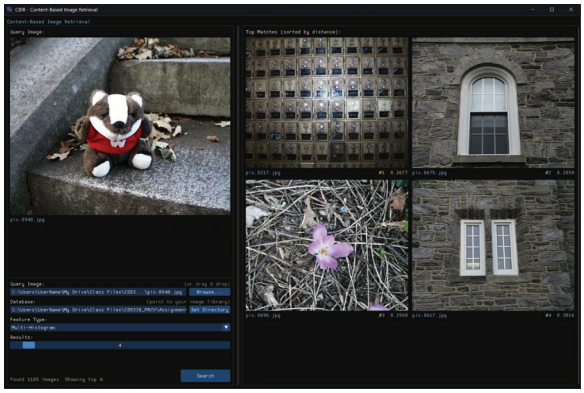
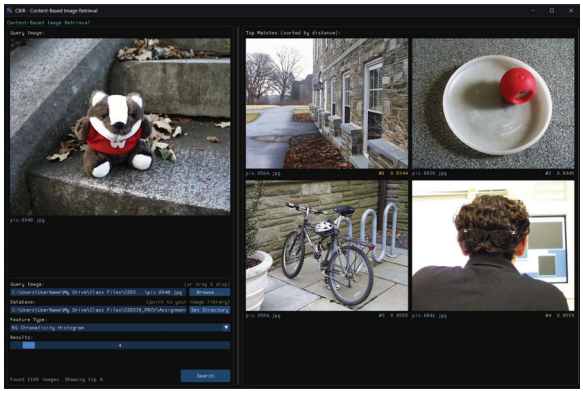
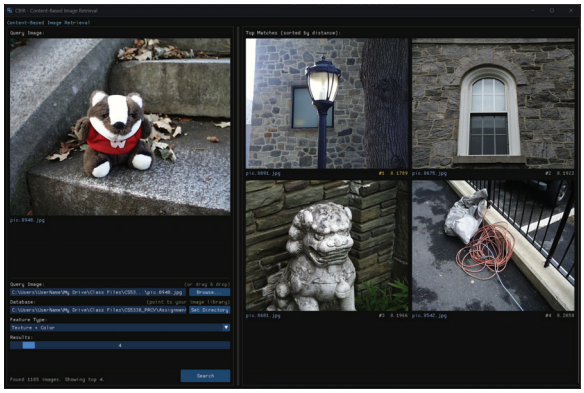

Description: The DNN embeddings's top results find the same subject of the query image, whereas the other results from the previous methods produce inconsistent matches.

Task 6: Compare DNN Embeddings and Classic Features

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

The search results for Pic.0948.jpg, Pic.0948.jpg and Pic.0734.jpg are in the following pages.

Pic.0948.jpg

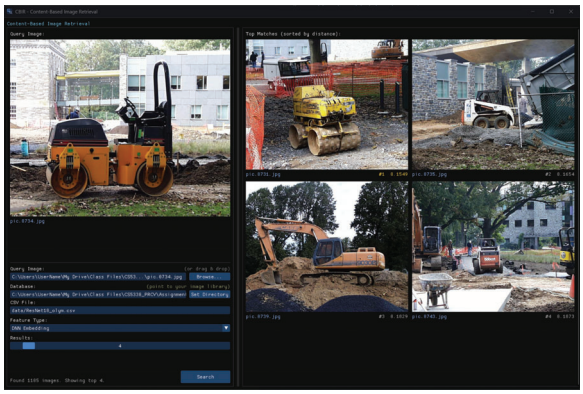
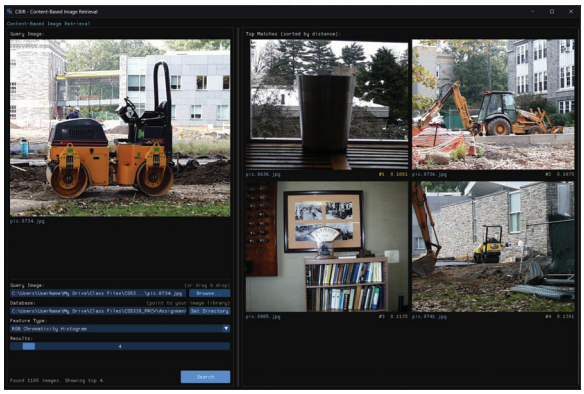
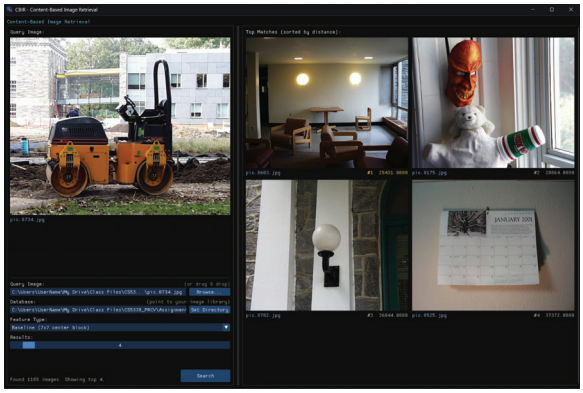
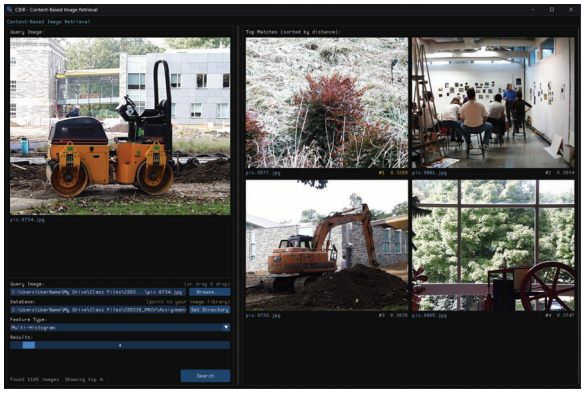
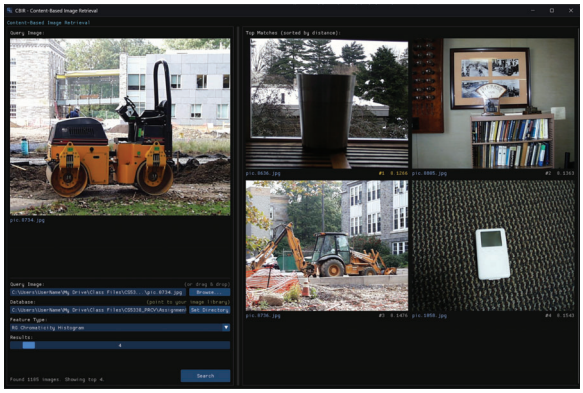
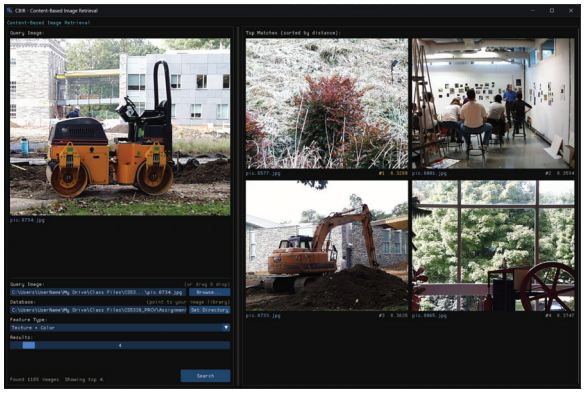







Description:

Pic.0948.jpg (badger stuffed animal):

The DNN embeddings returned 4 pictures of the same badger toy taken in different places, which shows that it knows what the object is. On the other hand, the classic features matched things that had nothing to do with each other. The baseline method returns back a dartboard and a glove, the histogram methods gave back things like a red bowl and a bicycle, and the texture+color method, stone buildings and a lion statue. This is because they only captured low-level properties like the gray tones of the sidewalk or the red/white color of the badger toy's outfit, not the subject's identity.

Pic.0734.jpg

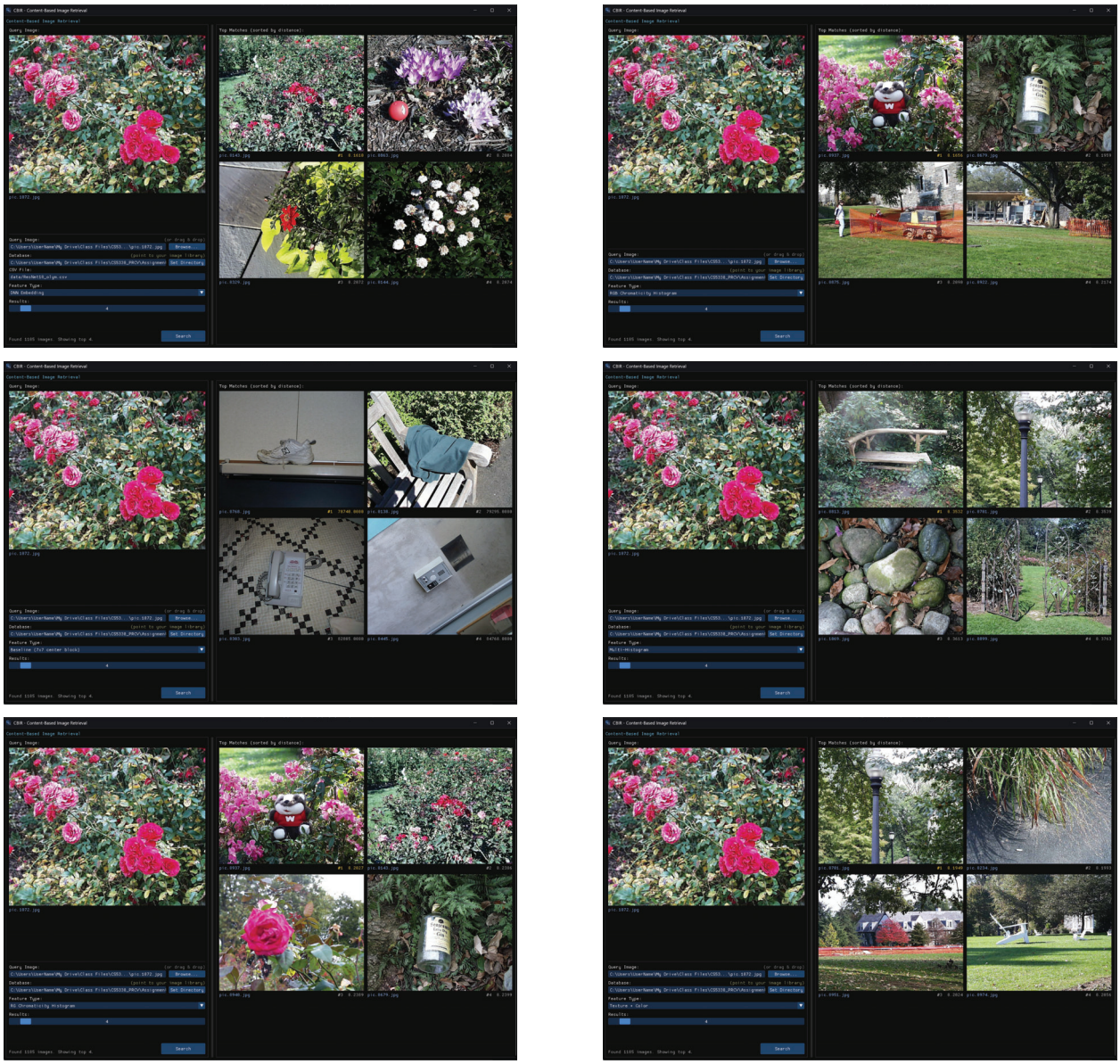







Description:

Pic.0734.jpg (construction excavator):

The DNN embeddings once again correctly identified the subject and returned other photos of the same construction site and heavy machinery. Most of the classic features didn't work. The baseline matched a mask and a lightbulb, the RG histogram matched a bookshelf, one other excavator, and an iPod, and the multi-histogram matched indoor gallery scenes and a wagon wheel. This is because they focused on the earthy brown/yellow color palette or gray textures of the dirt and machinery instead of realizing that the picture shows construction equipment.

Pic.1072.jpg



Description: Similar observation and results from *Pic.1072.jpg*.

In summary, the DNN embeddings can consistently find images that are semantically similar (of the same object or scene type) because they store high-level visual concepts learned during training. The classic features, on the other hand, only match surface-level attributes like color or texture, which means that the results look similar in pixel numbers but have nothing to do with each other in real meaning.

Task 7: Custom Design

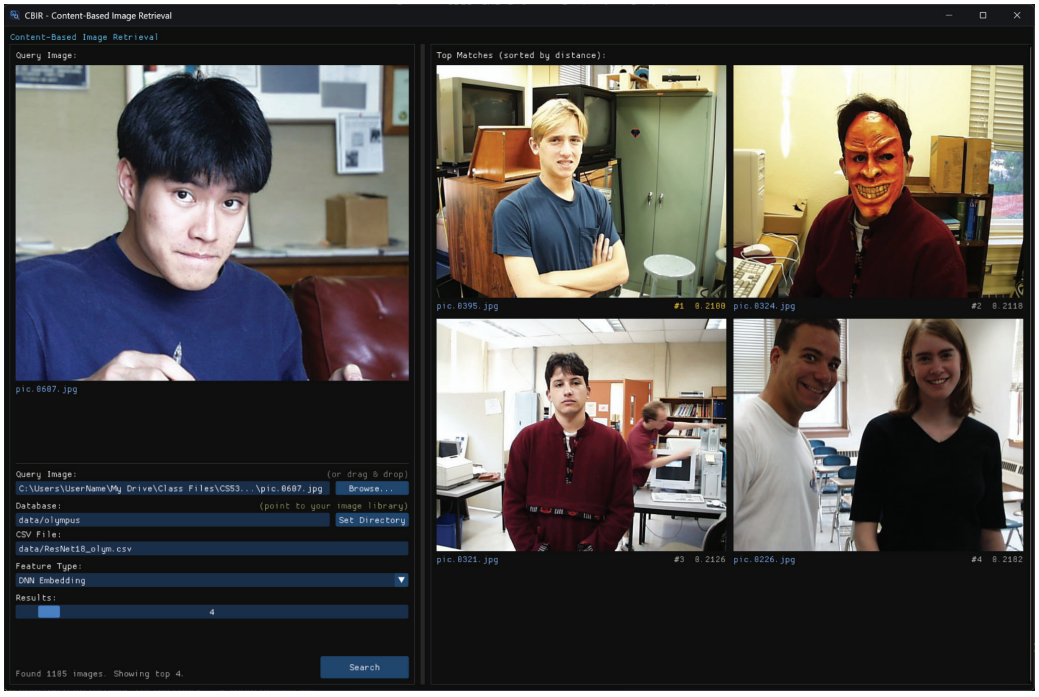
I designed a custom feature specifically for finding similar portrait images by combining:

- DNN embedding (70% weight) - Recognizes people and indoor scenes
- Skin tone histogram (20% weight) - Detects skin-colored pixels in the center region
- Center brightness (10% weight) - Matches lighting conditions

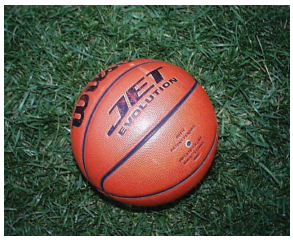
Query Image: pic.0607.jpg

Top 3 similar images:

- pic.0321.jpg (distance: 0.196799)
- pic.0226.jpg (distance: 0.203713)
- pic.0088.jpg (distance: 0.213427)

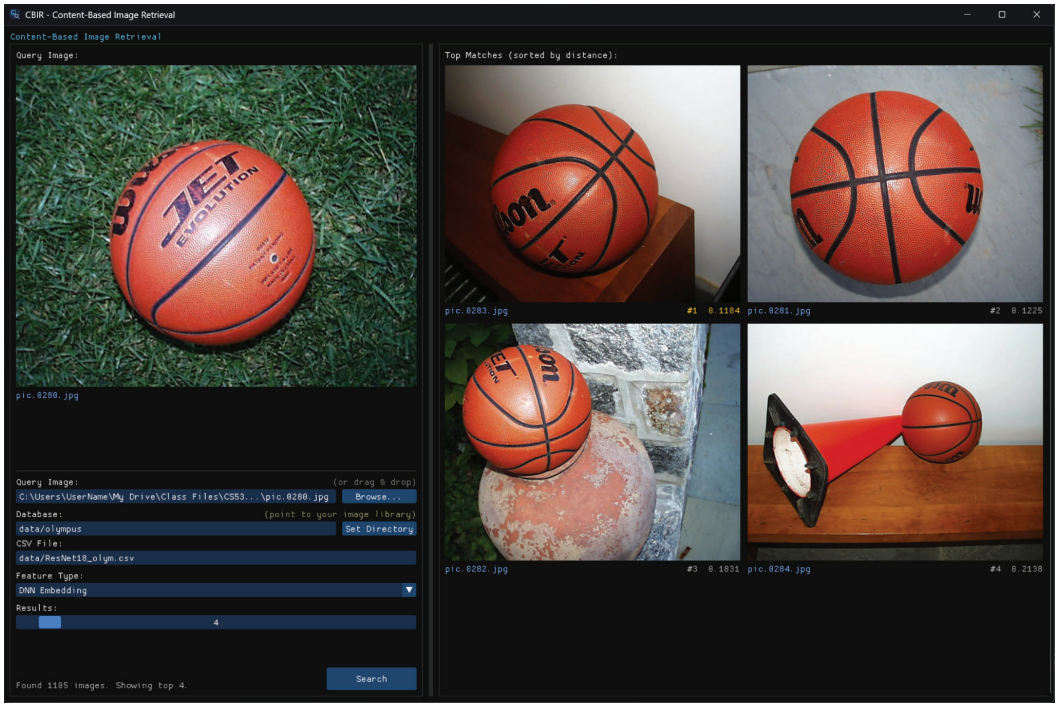


Query Image: pic.0280.jpg



Top 3 similar images:

- pic.0281.jpg (distance: 0.099395)
- pic.0283.jpg (distance: 0.107439)
- pic.0282.jpg (distance: 0.136391)



Description:

The custom method worked pretty well for finding similar portraits. The DNN part picks up on the person and that it's indoors, and the skin tone thing helps find other pictures with faces in the middle. The brightness matching works for indoor lighting too. All three matches probably have the same stuff - faces in the center, taken indoors, similar skin tones.

It's kind of funny that the custom portrait method still found really close matches even though this obviously isn't a portrait. The distances were super low (0.09-0.13) so these are probably just more photos from the same shoot, maybe the same basketball from different angles. I think the orange basketball probably got picked up as "skin tone" because of the HSV range I used

(H: 0-50), which accidentally helped it match similar colored objects. So even though I made this for portraits, it still works for other stuff if the colors and composition are similar.

The weighted combo I used (70% DNN, 20% skin/color, 10% brightness) definitely works for what I wanted with portraits, but it also picks up other centered objects with warm colors. The DNN being weighted so high keeps things semantically similar and the other features just help narrow it down more.

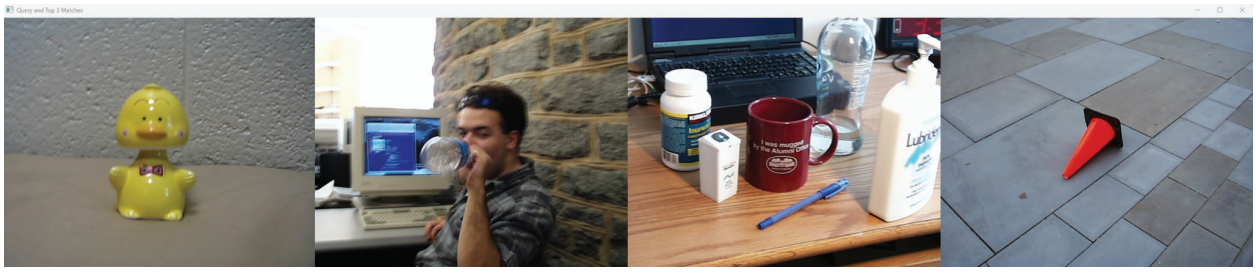
Extension: Additional Features- Oriented Gradient Histogram

I added a gradient orientation histogram that looks at which direction edges are pointing, not just how strong they are. The Sobel magnitude from Task 4 only tells you if there's an edge, but this tells you if it's vertical, horizontal, diagonal.

Query Image: pic.0462.jpg

Top 3 similar images:

- pic.0089.jpg (distance: 0.034325)
- pic.0637.jpg (distance: 0.041339)
- pic.0278.jpg (distance: 0.054249)



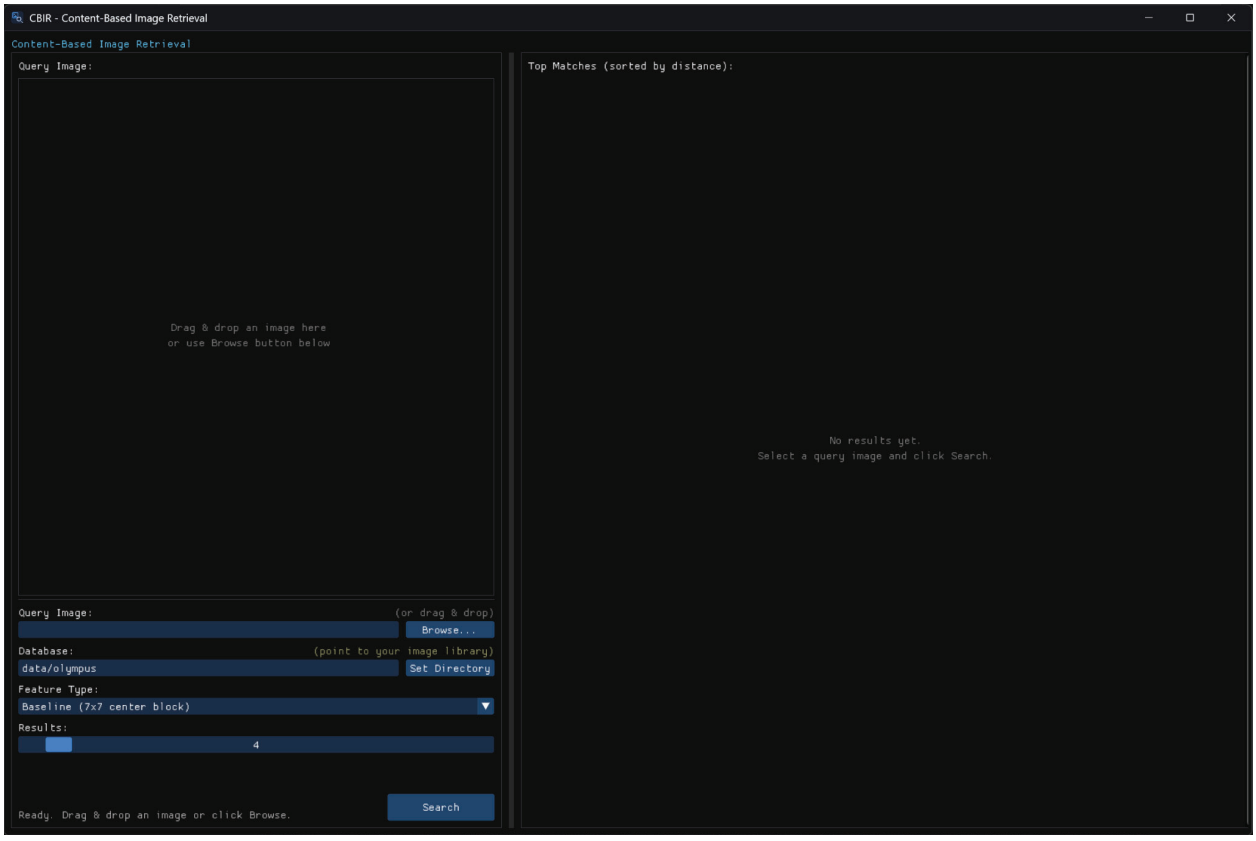
Description:

The gradient orientation finds images with similar edge patterns and ignores the colors. Unlike the texture and color method, it found things that looked similar in both structure and color.

Gradient orientation is way faster because it's so small. But it misses color completely, so you might get false matches. It would probably work better for stuff like building photos or drawings where structure matters more than color. For normal images where you care about both, Task 4 is the better option.

The oriented gradient histogram captures something Task 4 doesn't like the actual edge directions. But on its own it's not as good because it's too simple. It would probably work better if you combined it with other features or use it when you specifically care about structure and not color.

Extension: GUI with Dear ImGui



It was important to build a GUI with Dear ImGui so that we could iterate more quickly when trying out different types of features and distance metrics. Instead of running command-line queries over and over, we can now quickly switch the target image, change the image database directory, and see ranked results side by side. The interface also supports drag-and-drop for loading query images and keyboard shortcuts for common actions, making the overall workflow much smoother. See short video demo in the link below: https://northeastern.zoom.us/rec/share/mYpLk5Zslzz4vnWMtGBuKarEVmgLtT-6Uh7DYGw_VKNz230EhtIZKxhT2sEeibJC.a04ztKpc70vXcNFW?startTime=1770697946000

Passcode: **i*.ay2W?**

We have also made a release version on Github for Windows. See it in our repo here: https://github.com/ParkerCai/PRCV-Project2-Content-Based-Image-Retrieval/releases

Reflection:

Parker:
Working on the foundational tasks like baseline matching and histogram matching helped me understand how even simple features can produce useful retrieval results, as well as where they don't work.

Implementing cosine distance for the DNN embeddings in Task 5 was a nice contrast because suddenly the system could match on semantic similarity rather than just raw pixel values or color distributions. It felt like the retrieval system suddenly gained some wisdom. I ran into some difficulties parsing and retrieving the embedding data from the CSV file, and ended up optimizing it by building a hash map for lookups instead of looping through every row each time.

Building the GUI with Dear ImGui was the most rewarding part. It turned a boring command-line workflow into something interactive where I could quickly test different feature methods and see how they worked. This makes the entire system more real and closer to being a useful tool, as well as making debugging and testing the other tasks much faster.

Jenny: The biggest thing I learned is that there's no single "best" way to compare images. Different features work better for different situations. Like, color histograms are great for finding similar colors but terrible if you care about what the actual objects are. I was surprised by how much feature choice matters. When I tested pic.0535 with different methods, I got completely different results every time and I needed to think about what kind of similarity I actually want before picking a method.

For the custom design (Task 7) was the most interesting part. I had to actually think about what makes portraits similar, like not just colors or edges, but specific things like skin tones and face positions. Combining the DNN with skin detection and brightness felt like I was building something useful, not just following steps. It was cool to see it work, and even cooler that it accidentally worked on the basketball photo because of the orange color. The hardest thing for me was building the system and getting the configuration to work without any errors. Overall, this project changed how I think about image search.

Acknowledgements:
AI tools, including Gemini and Claude, were used to set up the VSCode development environment for C++, CMake, OpenCV, CUDA, and other technologies. Additionally, Claude Opus 4.5 assisted with code debugging and helped me learn and implement most of the ImGui frontend code.