Adithya Palle

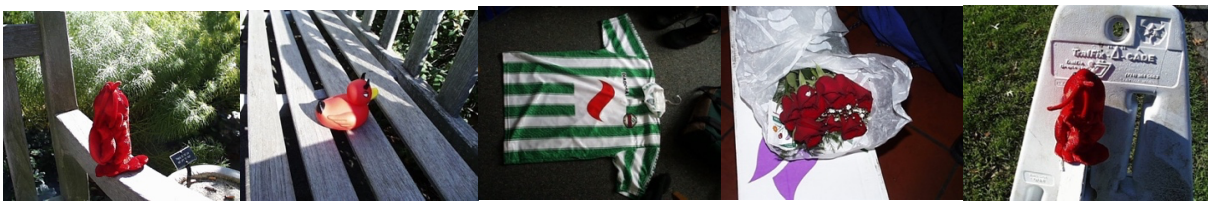
Content-Based Image Retrieval

Overview:

In this project, we explore various feature extraction methods and distance metrics for content-based image retrieval (CBIR) algorithms. We develop a flexible pipeline that allows mixing and matching different feature extraction techniques and distance metrics, conducting multiple experiments to evaluate their effectiveness in different CBIR tasks. We begin with simple center crops and SSD distance metrics, then explore color histogram-based approaches. Next, we investigate texture-based methods using Sobel gradients before delving into deep network embeddings. We also experiment with combining depth information with RGB histograms for enhanced matching. Additionally, we examine the viability of applying Fourier transforms to Sobel gradient images for matching objects with uniform, well-defined patterns, such as solar panels.

Baseline Matching:

Here, we find similar images using the sum of squared distances as a distance metric on the 7x7 center square of the images. My top 4 matches for pic.1016.jpg were pic.0986.jpg, pick0.641.jpg, pic.0547.jpg, and pic1013.jpg.



Histogram Matching:

For my histogram feature extractor, I decided to extract a whole RGB histogram using 8 bins for each of the channels, and I used histogram intersection as the distance metric. I went with the choice of a 3D histogram because I felt that the brightness/intensity features which are lost in 2D chromaticity diagrams are important in matching images. Below are my top 4 matches for pic.0164.jpg:

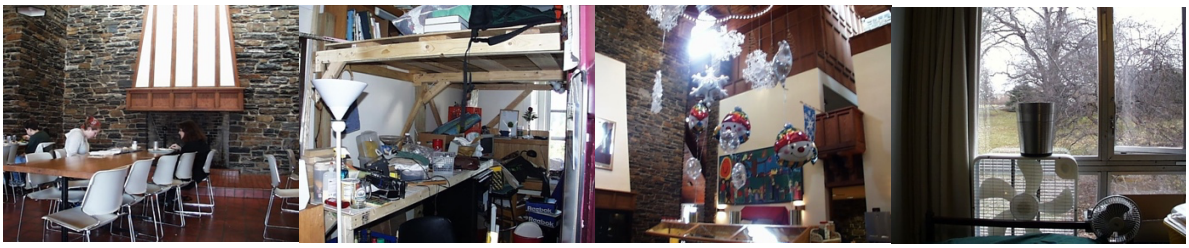
Adithya Palle

Multi-Histogram Matching:

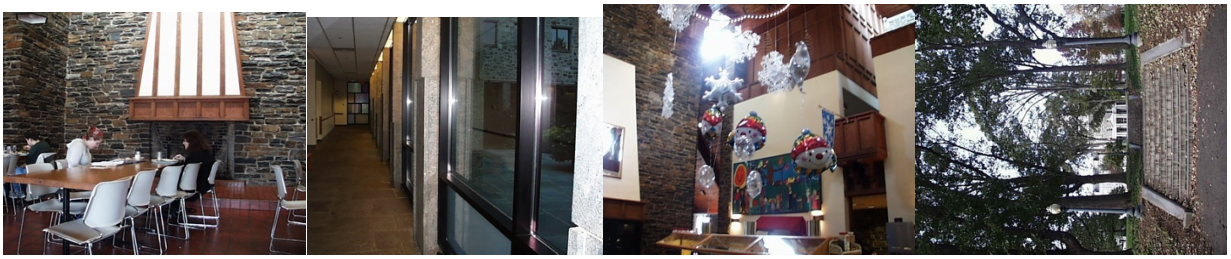
I decided to use a 8-bin red and green chromaticity histogram (2D) and a 3D RGB histogram and equally weight the similarities with a simple sum to get a multi-histogram matching algorithm. Below are my top 4 matches for pic.0274.jpg:

Texture and Color:

I decided to use a Sobel X and Y magnitude 3D histogram as a texture feature, and combined it with the 3D RGB color histogram by summing the separate histogram intersections to get the below top 4 matches for pic.0535.jpg:



Here are the top 4 matches I get when using a single 3D RGB histogram with histogram intersection:



Here are the top 4 matches when I apply the multi-histogram method (2D chromaticity histogram and 3D RGB histogram):

Adithya Palle

Since all the methods have some contribution from the 3D RGB histogram, it's not surprising that the matched results are similar. One thing to note is that the texture method seems to only match with images that have many details such as tree branches in the last match or the items on the desk in the first match, and it does not match with images that have fewer textures such as the hallway image that the other methods match with first. This is possibly because of the gradient magnitude to pick up on the many edges involved with these details and match it with the many edges in the rocky wall in the background of the main image.

Deep Network Embeddings:

Below are my top matches for pic.0893.jpg using the pre-calculated ResNet18 embeddings:

Here are my top results for other methods

Texture + RGB

RGB + Chromaticity histograms

Adithya Palle

RGB Histogram

From these results, we can see that the ResNet embedding matching algorithm was the only one that returned images exclusively of fire hydrants in its top matches. This demonstrates the ability of embeddings to derive semantic understanding from images, enabling them to retrieve matches that better align with human perception.

Compare DNN Embeddings and Classic Features

Below I compare all of the above methods for pictures 1072, 948, and 734. The images are ordered from top to bottom in ascending distance, meaning the top image is the best match.

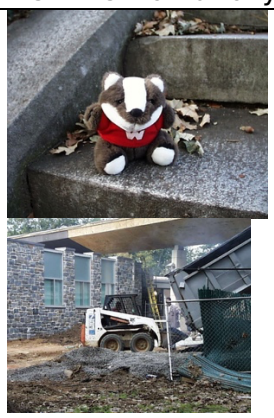
For the below tables, the top image in each column is the target image.

Pic 1072

ResNet	RGB + Chromaticity	RGB	Texture + RGB

Adithya Palle

Pic 948

ResNet	RGB + Chromaticity	RGB	Texture + RGB
			

Adithya Palle

Pic 734

ResNet	RGB + Chromaticity	RGB	Texture + RGB

From the above examples, the ResNet matching algorithm excels at retrieving matches where the primary objects are the same. In this sense, deep network embeddings are highly

effective for "object matching." However, this does not necessarily mean they are the best for overall image matching.

According to the assignment details, these embeddings were generated from a network trained on ImageNet, a dataset that focuses on describing the primary object in an image. As a result, ResNet embeddings emphasize object recognition while placing less importance on other visual details, such as color and texture in the background.

This can be observed in image 1072, where color-based methods retrieve a thematically fitting image of a plushie sitting in a bush with pink flowers, while ResNet misses this match entirely because the primary object does not align. Similarly, in image 948, the texture-based approaches retrieve images with stone backgrounds that match the plushie on a sidewalk, whereas ResNet focuses solely on the plushie, disregarding the stone texture—despite it being a prominent feature.

While ResNet embeddings trained on ImageNet are powerful for object-level similarity, they often overlook background details and secondary objects, making them less reliable for holistic image matching. However, if a deep network were trained on an image annotation dataset that captures all aspects of an image—objects, background, textures, and colors—it could potentially produce embeddings that consider multiple visual factors and yield a more comprehensive image matching algorithm.

Custom Design

For this task, my goal is to create a feature extraction method and a distance metric to match dinosaur images with other dinosaur images. For my feature extraction method, I used DepthAnything to determine the depth value for each pixel and created a foreground mask by ignoring all pixels (set to Magenta) above a certain distance threshold (0.5). I then created a RGB histogram from this foreground (by ignoring the magenta pixels). For the distance metric, I implemented an intersection over union metric which is calculated with the following formula:

$$D = 1 - \frac{\sum \min\big(H_1(i), H_2(i)\big)}{\sum\big(\max\big(H_1(i), H_2(i)\big)\big)}$$

Here, $H_1(i)$ represents the i-th pixel (assume the histogram was flattened) of the first histogram, and $H_2(i)$ is the same for the second histogram. This is a loss function used in segmentation tasks, but I noticed its similarity to histogram intersection and wanted to see if it was a viable histogram comparison method as well.

Below, I present the results for two dinosaur images that were matched against a modified Olympus dataset that included 10 additional dinosaur images. The left image is of a dark-

orange dinosaur, while the other has a redder hue. Since the other dinos in my dataset match the color of the orange dinosaur, I would expect the orange dinosaur to match with other dinosaur images, but the red dinosaur image may match with other images particularly those with red colors.

Top Matches (ascending by distance):

Orange Dinosaur (top image is target)	Red Dinosaur (top image is target)
	

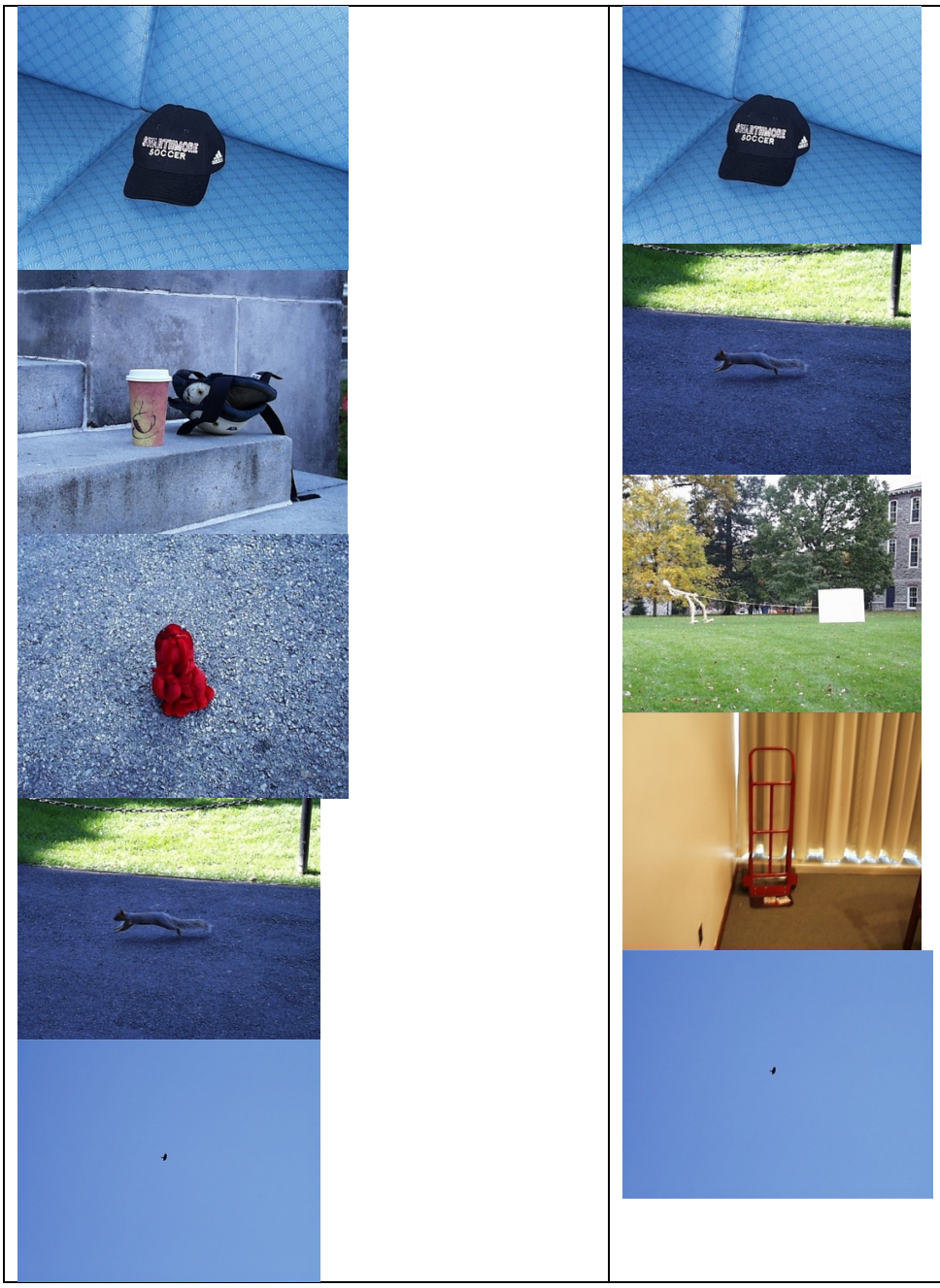
Adithya Palle

Worst Matches (ascending by distance):

Orange Dinosaur	Red Dinosaur

Adithya Palle



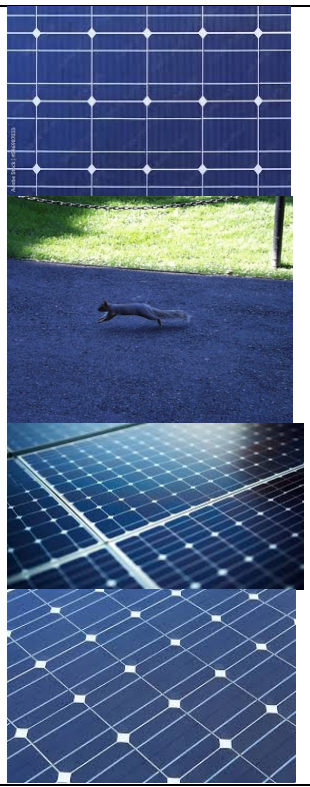
As shown above, the orange dinosaur had 3 out of 5 top matches as dinosaurs, whereas the red dinosaur had none. This underscores the limitations of RGB histograms for object matching in images, even with foreground segmentation. However, achieving a 60% dinosaur match rate is notable, given that only 10 out of 1010 images in the modified dataset were dinosaurs. Without foreground segmentation using DepthAnything, the results would likely be even more inconsistent due to the diverse color spectrum in the dataset.

Adithya Palle

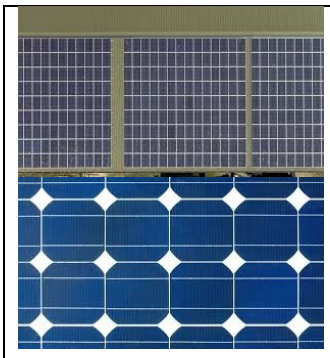
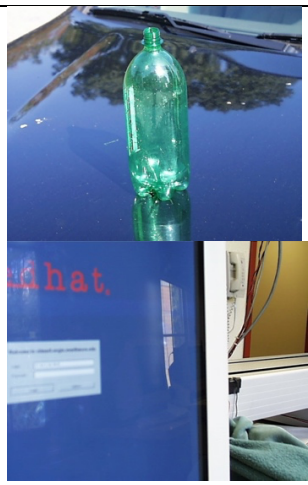
Extension – Fourier Transform for matching solar panels

Inspired by a project Professor Bruce Maxwell mentioned in class, I seek to attempt to use Fourier Transforms to match images of solar panels with other solar panels. Solar panels are manufactured to have uniform, distinct line patterns on them that are wave-like and would produce notable energies on the FFT spectrum. First, to remove noise due to finer details in the image, I take the Sobel gradient magnitude image of the original image to only highlight the key edges, then I run the FFT algorithm on it to produce a log-scaled FFT magnitude image. Since I am interested in the more frequent patterns which are present on solar panels, I also black out the center square of the image to effectively ignore low frequency signals. I use SSD as the distance metric to match images with similar frequency peaks in the FFT spectrum. Since SSD effectively captures spatial differences in color intensity, spectra with peaks at different frequencies will result in intensity variations at different locations in the spectrum image, which SSD can detect. Below, I present the top matches for images of solar panels compared with other methods, where the top image in each column is the target image:

<u>Solar Panel 1</u>

Sobel + FFT + SSD	RGB + Chroma	Texture + RGB
		

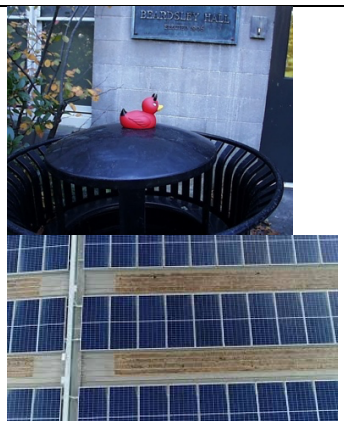
Adithya Palle

Solar Panel 2

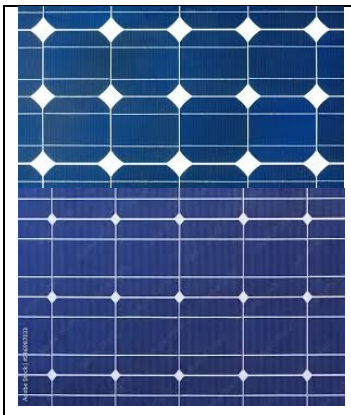
Sobel + FFT + SSD	RGB + Chroma	Texture + RGB
		

Adithya Palle

Solar Panel 3

Sobel + FFT + SSD	RGB + Chroma	Texture + RGB
		



These results are surprisingly good, but they came after several failed attempts. My initial iterations lacked the Sobel gradient, used the entire FFT, and combined the FFT image with an RGB histogram using a joint distance metric. However, these approaches produced inferior results compared to what I present here.

The key strength of this approach is its robustness against noise and minor texture variations, thanks to the Sobel gradient and the masking of the FFT image to emphasize high-frequency regions. Additionally, by avoiding color features, this method effectively matches images of solar panels with varying hues—such as the third solar panel image, which performed particularly poorly with color-based methods.

This idea could be extended to matching other images and objects with uniform, well-defined patterns, such as fences, honeycombs, and similar structures.

Reflection

This project deepened my understanding of both classical and modern image matching approaches. I now have a stronger grasp of the fundamentals of feature extraction and distance metrics and how they can be applied to create both simple methods and advanced deep learning-based techniques.

One key takeaway was that classical methods can sometimes outperform deep-learning approaches, as seen in the object retrieval task where the embedding-based method overfitted to the primary object in the image. While AI-based methods will likely dominate future advancements in this field, I now believe classical approaches remain valuable, especially in data preprocessing and feature generation for deep-learning models.

My favorite part of the project was extending it by applying the Sobel gradient approach in combination with Fourier transforms, as we learned in class, to detect objects with uniform, wave-like patterns. The effectiveness of this algorithm in matching solar panels highlighted the practical power of such techniques.

Lastly, this project significantly tested my ability to write clean, extensible code and build infrastructure for computer vision pipelines. A major requirement was ensuring a modular design that allowed easy integration and testing of different feature extraction and distance metric methods. Overall, I feel like I've grown as a computer vision programmer and engineer through this experience.

Acknowledgement

For this project, the sources I consulted were Bruce Maxwell's code demos and lectures, Chapter 8 of *Computer Vision* by Shapiro and Stockman, OpenCV documentation, and a couple of Stack Overflow posts which were helpful in debugging certain implementation issues.