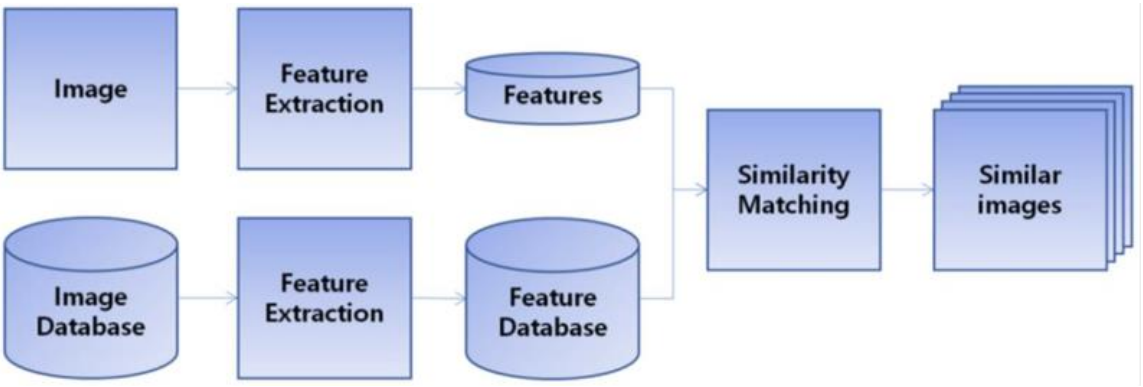
PATTERN RECOGNITION AND COMPUTER VISION

Project 2

BY: Veditha Gudapati & Tejasri Kasturi

INTRODUCTION:



CBIR systems has two main features:

Global: Global features are those that describe an entire image. They contain information on the entire image. For example, several descriptors characterize color spaces, such as color moments, color histograms, and other visual elements such as e.g. shapes and texture.

Local: Local features describe visual patterns or structures identifiable in small groups of pixels. For example, edges, points, and various image patches.

This project, which was originally intended to be a command-line tool, has a user-friendly graphical user interface that allows users to choose a directory that contains an image database and enter the target picture filename. By choosing the feature analysis and matching approach, users may explore picture matching and pattern recognition using an intuitive numpad-based interface. The application then uses this input to find and display the top N images in the database that most closely match the target image, providing an engaging and instructive experience in the field of image analysis.

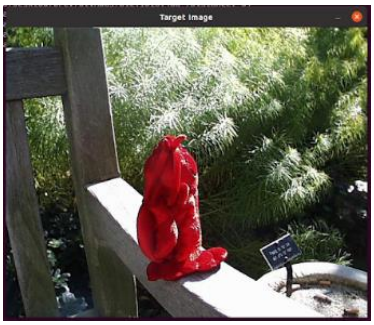
TASK 1: BASELINE MATCHING

The core task of this project is to navigate a database of images and pinpoint those that exhibit content similarity to a specified target image. The approach incorporates the utilization of classic features and deep network embeddings. To execute the task, have opted for key presses instead of running on the command line.

According to the code, we use feature extraction method and distance metrics to find similar images in a given directory. Here, **computeBaselineFeatures** is a function that extracts baseline features from an input image. The baseline feature is a 7x7 pixel square taken from the center of the image. It extracts this region of interest (ROI) and flattens it into a one-dimensional row vector (`reshape(1, 1)`).

In the `main` function, when keypress ‘b’ is presses this feature is computed for both the target image and each image in the directory. Then, it calculates the L2 norm (Euclidean distance) between the baseline features of the target image and each image in the directory

We are using feature extraction and distance metric, feature extraction is based on a 7x7 pixel taken as base feature and matched with images with similar color and texture with overall database of images. Distance metric is used to find similarity and dissimilarity between feature vectors. The sum-of square distance should be at least to find the best accuracy.



Target Image

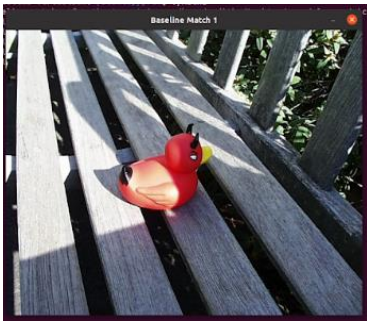


Image pic.0641.jpg

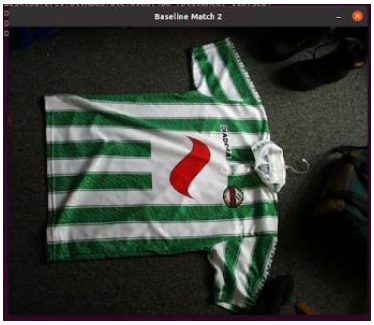


Image pic.0986.jpg

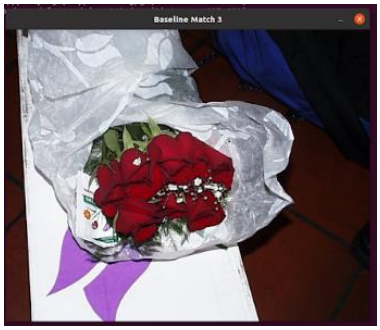
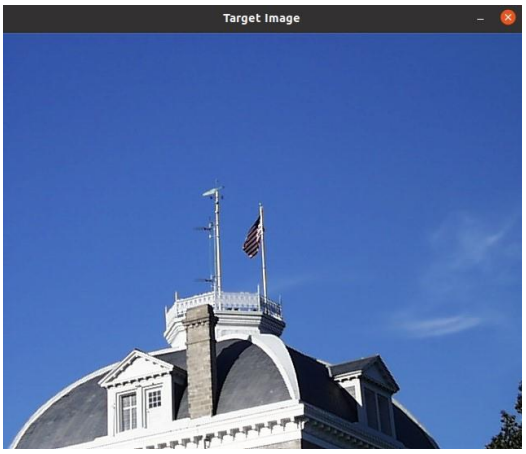


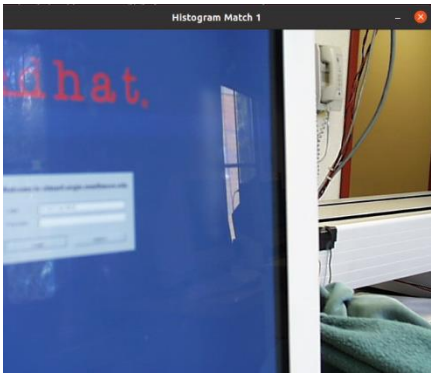
Image pic.0547.jpg

TASK 2: HISTOGRAM MATCHING

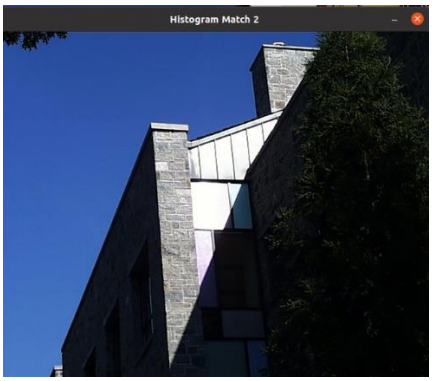
In the second task we are using the HSV color space to compute the histogram features for an input image. First, the cvtColor() function from the OpenCV library is used to transform the image from the BGR color space to HSV. The hue (H) and saturation (S) channels of the HSV picture are then used to compute the histogram. Each channel's histogram is calculated independently using the calcHist() method. The level of detail of the histogram representation is decreased by setting the number of bins for the H and S channels to 8. The S channel has a range of 0 to 256 degrees, and the H channel's range is set at 0 to 180 degrees. The binning scheme of the histograms is defined by these parameters. After computing the histograms, they are normalized to ensure consistency in feature scaling across different images. Finally, the histogram is reshaped into a 1xN matrix before being returned as the computed histogram features of the input image.



Target Image



pic.0080.jpg



pic.1032.jpg

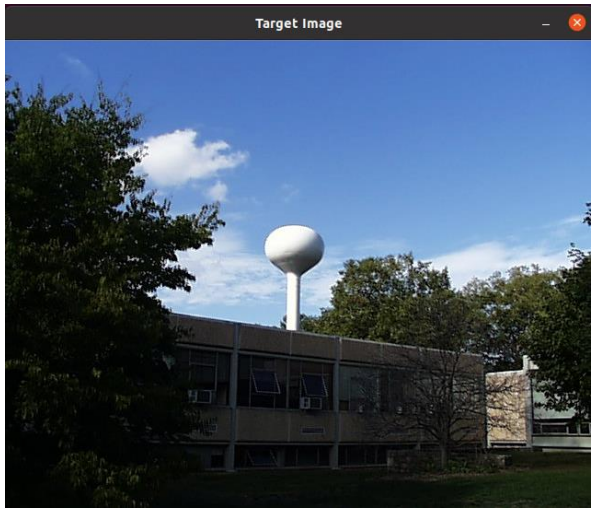
pic.0110.jpg

TASK 3: MULTI-HISTOGRAM MATCHING

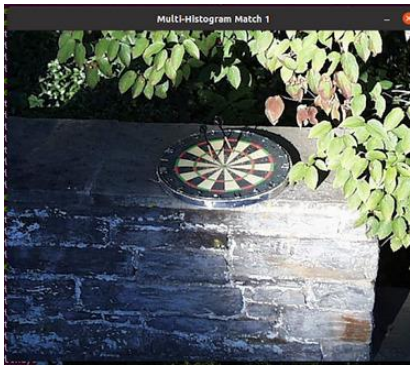
For the multi-histogram matching in the provided code, two separate histograms are used:
Whole Image Histogram which represents the entire image and Center Region Histogram that uses only the center region of the image.

For each histogram:
The H (hue) channel is divided into 16 bins.
The S (saturation) channel is divided into 16 bins.
Therefore, each histogram individually has a total of 16 x 16 = 256 bins.

In this custom distance metric, when ‘f’ is pressed both hist1 and hist2 are the histograms of the two images being compared. compareHist() function with HISTCMP_INTERSECT method calculates the histogram intersection between them. Then, the custom distance metric computes the weighted average distance using the histogram intersection values. The weights 0.6 and 0.4 used in the weighted averaging can be adjusted based on the application requirements.

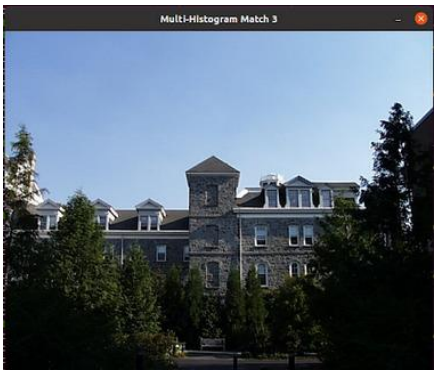


Target Image



pic.0156.jpg

pic.0431.jpg



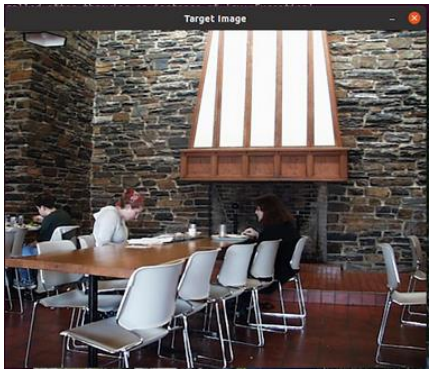
pic.0213.jpg

TASK 4: TEXTURE AND COLOR MATCHING

For this task we are dividing into texture matching and colour matching. For texture matching whole image is used as reference to calculate using Sobel magnitude image, which is generally helpful here as it detects edges, magnitude of the image which thereby increases strength of the final image matched. According to Our understanding the flow is such that when target image is taken it is converted to greyscale and gradient magnitude is applied to which Sobel magnitude image to create a histogram of gradient magnitudes which gives a texture feature. Which is combined with colour feature matching to give images of feature and colour top matches.

While sorting the distance a distance metric that weights the two types of histograms (color and texture) equally. This means that when comparing two images, both the color and texture information should contribute equally to the distance calculation, which is achieved by defining a suitable distance measure that combines the distances for color and texture.

Target Image



pic.0535.jpg

Texture and color feature matching images.

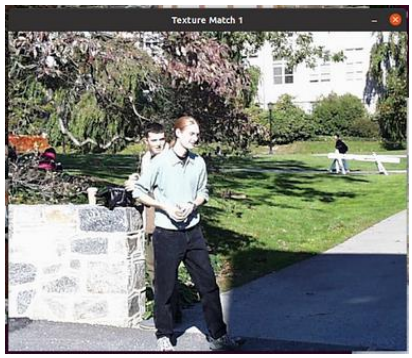


Image 1

Image 2

Image 3

TASK 4 (EXTENSION)

1. Laws Texture matching:

   The filter2D function is used to apply the masks of the laws (L5, E5, S5, W5, R5) on the grayscale version of the image. As a result, nine distinct convolutions are produced (L5E5, E5L5, L5L5, E5S5, S5E5, E5E5, S5S5, L5S5, S5L5). Each of these convolutions' energy values is calculated using the norm function and NORM L2 (Euclidean norm). Several facets of the texture information are represented by these energy values. For every image, a feature vector is created by concatenating the energy values obtained from the convolutions. The energy values of the nine convolutions are contained in a 1x9 matrix that makes up this feature vector. Laws' masks are used to compute the feature vectors for each pair of images (the target image and the comparison image).

   We are computing the Euclidean distance (L2 norm) between the feature vectors of

Histograms of Laws filter responses feature matching images.

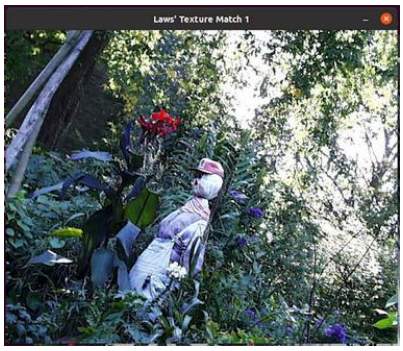


Image 1

Image 2

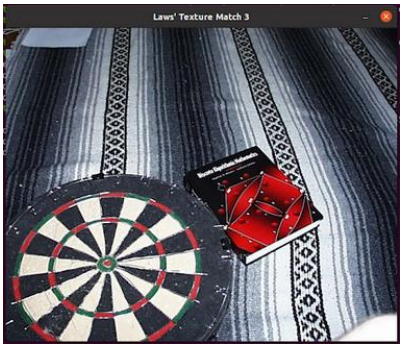


Image 3

2. Gabors' Histogram:

The parameters kernelSize, sigma, theta, lambda, gamma, and psi are used to define gabor filters. There are eight Gabor filters produced, ranging in orientation from 0 to 7 in π/4 steps. Each image's feature vector is created by concatenating the responses from the Gabor filter horizontally. This feature vector is represented as a 1x(N*kernelSize) matrix, where kernelSize is the Gabor filter's size and N is the number of Gabor filters. When we use the 'g' key, the top N matching photos based on Gabor filter images are displayed.
This method attempts to produce histograms by extracting Gabor filter responses and representing the textural properties of the images. Based on the separations between these histograms, the matching is carried out.

Histograms Gabors' responses feature matching images.

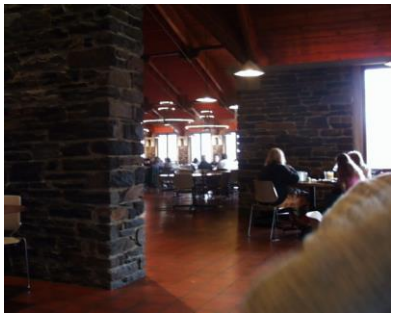


Image 1

Image 2

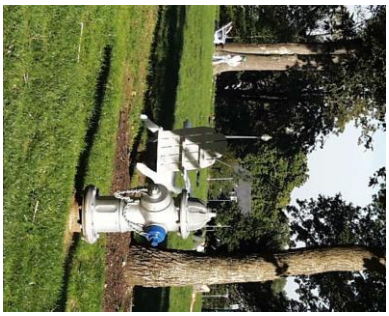


Image 3

3. Fourier Transformer:

The Fourier Transform is commonly used in image processing to analyze the frequency content of an image. In this case, the focus is on extracting features from the magnitude spectrum of the Fourier Transform, and the resizing step is performed to create a compact representation suitable for matching.
The flow of code is as follows The input image is converted to grayscale using cvtColor with the conversion code COLOR_BGR2GRAY. The grayscale image is then converted to CV_32F (single-precision floating-point) using convertTo.
This is a common step before applying Fourier Transform. The Fourier Transform is computed using dft (discrete Fourier transform) on the grayscale image.
The flag DFT_COMPLEX_OUTPUT indicates that the output should be in complex number format. The complex output of the Fourier Transform is split into real and imaginary parts using split. The magnitude spectrum is computed using the magnitude function.
It represents the magnitude of the complex numbers at each pixel. As for resizing the displayed image the magnitude spectrum is resized to a 16x16 image using resize. This is a step to reduce the dimensionality and create a compact representation of the Fourier Transform, it is then the

resized magnitude spectrum is normalized to ensure that the feature vector is scale-invariant. Next is the step where it is converted to 1-D vector and resized to 1x256 matrix.

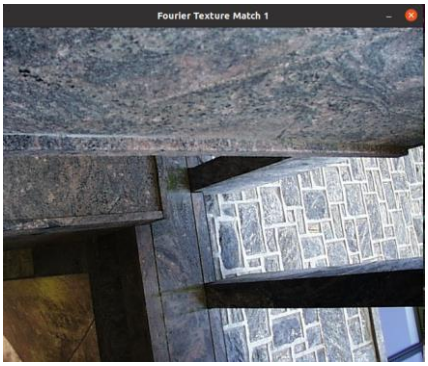


Image 1

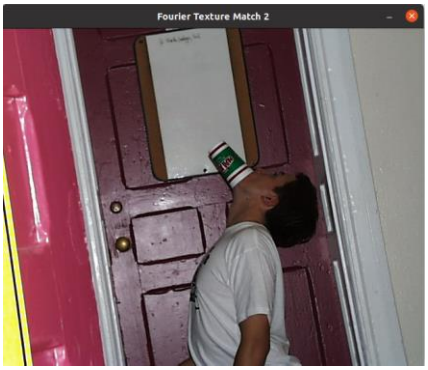


Image 2

Image 3

TASK 5: DNN EMBEDDINGS

In this task, the feature vectors provided in the CSV file represent the output of the final average pooling layer of a ResNet18 deep neural network pre-trained on ImageNet. The ResNet18 model is a convolutional neural network (CNN) architecture designed for image classification tasks.

Each row in the CSV file corresponds to an image. The first column contains the filename, and the subsequent 512 values represent the feature vector extracted from the last layer of the ResNet18 model. These feature vectors capture high-level representations of the images in a 512-dimensional space.

The goal of this task is to match or compare these feature vectors to find images that are similar or close in this high-dimensional space. Two distance metrics are suggested: sum-square distance and cosine distance.

Sum-Square Distance:
This is a straightforward distance metric where the distance between two vectors is calculated as the sum of squared differences between their corresponding elements.
For vectors v1=[x1,x2,...,xn]v1=[x1,x2,...,xn] and v2=[y1,y2,...,yn]v2=[y1,y2,...,yn], the sum-square distance is given by:

$$\sum i=1n(xi-yi)2\sum i=1n(xi-yi)2.$$

Cosine Distance:
Cosine distance is often used in high-dimensional spaces, especially in the context of similarity between vectors.
It measures the cosine of the angle between two vectors and is defined as:
$d(v1,v2)=1-\cos f_0(\theta)d(v1,v2)=1-\cos(\theta)$, where θ is the angle between the vectors.
The cosine of the angle between two normalized vectors v1v1 and v2v2 can be calculated as:
$\cos f_0(\theta)=v1\cdot v2\|v1\|\|v2\|\cos(\theta)=\|v1\|\|v2\|v1\cdot v2$.

The use of these distance metrics allows you to quantify the similarity or dissimilarity between feature vectors. In applications like image retrieval or content-based image search, such metrics help identify images that are close or similar to a target image based on their deep feature representations. The lower the distance, the more similar the images are considered to be. This task leverages the pre-trained ResNet18 model to provide meaningful and semantically rich representations of images for such comparisons.

Target Image

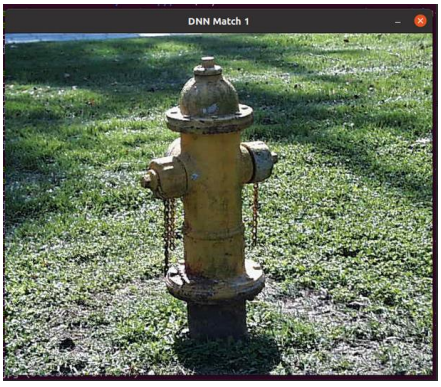


Image 1

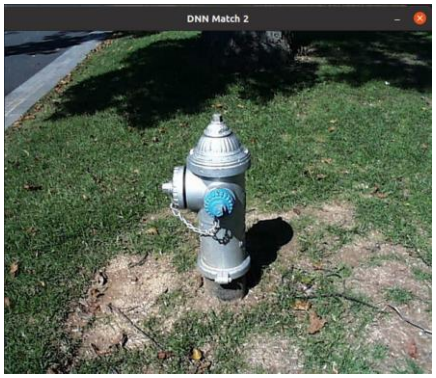


Image 2

Image 3

TASK 6: COMPARING CLASSIC FEATURES Vs DNN

As for the observation made on image (), we can conclude that DNN embeddings tend to capture high-level semantic information in the images, this might be that Features are learned automatically through the network's training process.
DNNs discover complex patterns and relationship and its Effective for recognizing objects, scenes, and abstract features. Whereas in classic features it is manually designed and is computationally efficient to compute compared to DNN embeddings.
DNN embeddings provided higher accuracy in tasks that require understanding complex visual patterns. Also one of the observation is that DNN embeddings require large amounts of labeled data for training, while classic features can be effective with smaller datasets.
The effectiveness of DNN embeddings often shines in tasks with large and diverse datasets, where the model can learn rich representations automatically. Classic features remain relevant, especially in scenarios where interpretability, simplicity, or limited data are important considerations.

Target Image

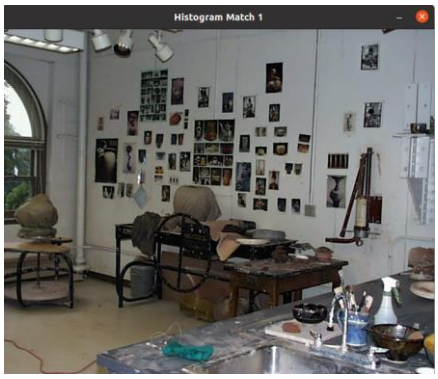


Image 1

Image 2

Image 3

DNN features

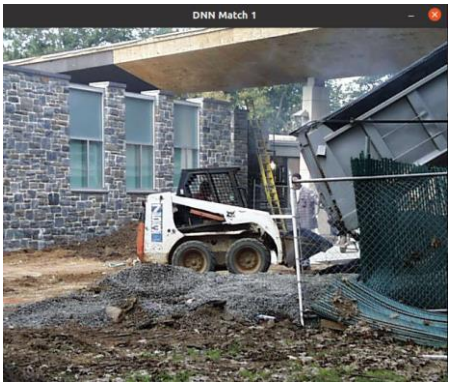


Image 2

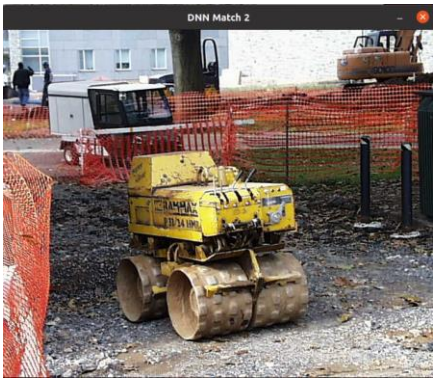


Image 2

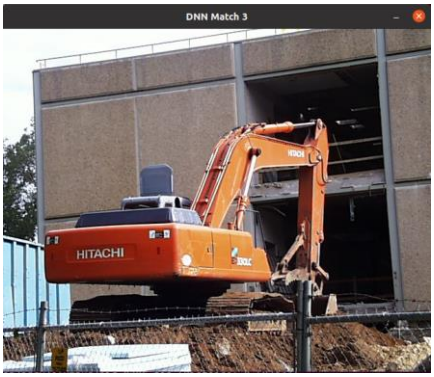
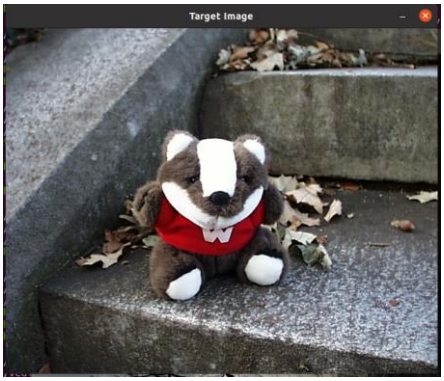


Image 3

Example-2



Classic feature(Histogram)

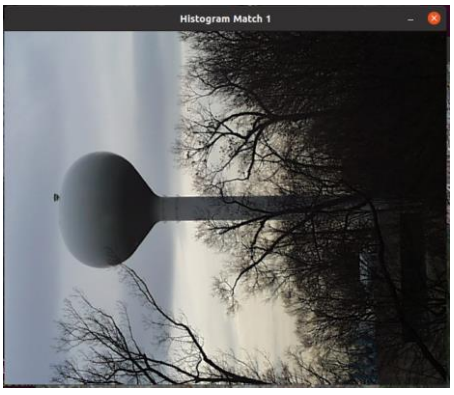


Image 1

Image 2

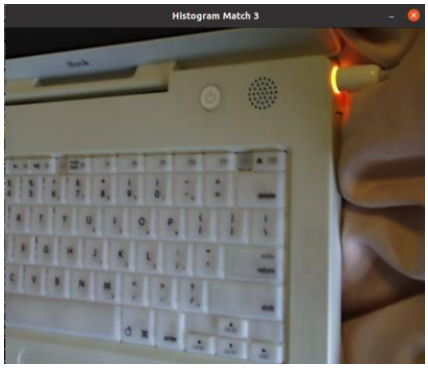


Image 3

DNN Features

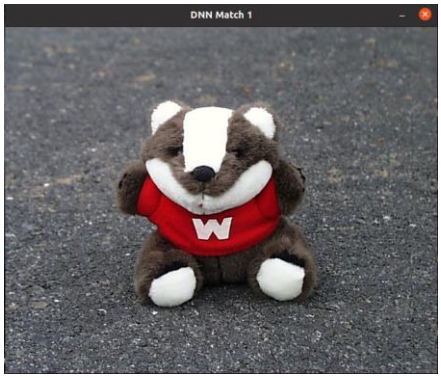


Image 1

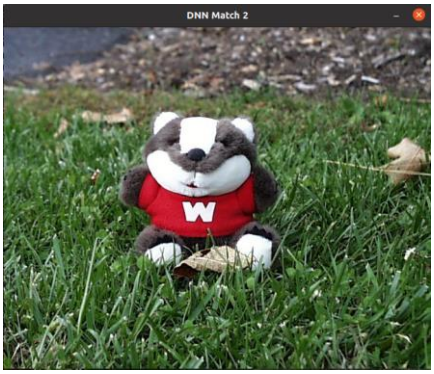


Image 2

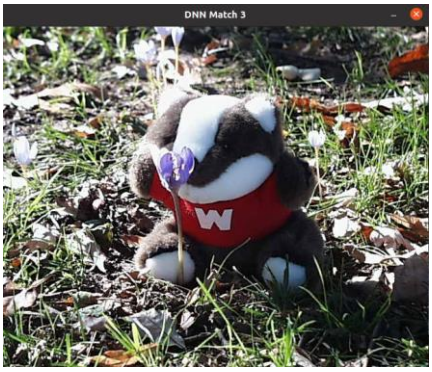


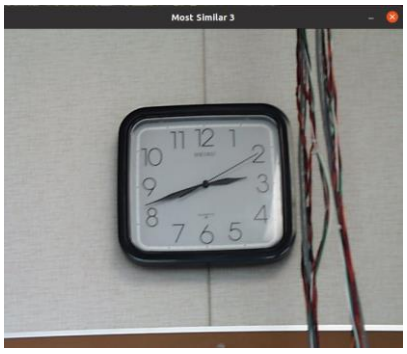
Image 3

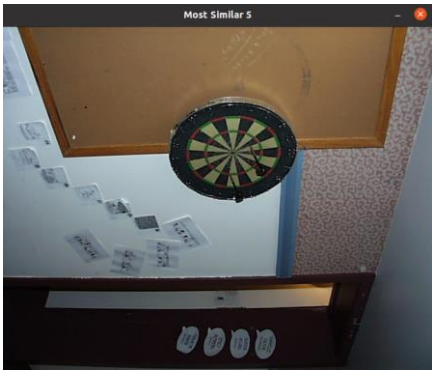
TASK 7: SHAPE AND FEATURE DETECTION

The function readFeatureVectorsFromCSV retrieves feature vectors from a CSV file. It extracts the feature vector by parsing each line and assuming that the filename is the first column. After that, each filename is connected to its matching feature vector via this vector being stored in an unordered map.
The cosine similarity between two feature vectors is calculated using the cosineSimilarity function. It computes the norms and dot product of each vector as it goes over each element in the vectors. After that, normalizing prevents division by zero and returns the cosine similarity.
The Sobel gradients of an input image are calculated using the computeSobelGradients function. The picture is converted to grayscale, Sobel operators are applied in both the x and y axes, the gradients' mean is determined, and the results are returned as a vector of floats.
The image retrieval system's fundamental component is the dnn_image_retrieval function. Using readFeatureVectorsFromCSV, it first reads the feature vectors from the CSV file. After that, it loads the user-specified target image and calculates its Sobel gradients. It determines the cosine similarity between the feature vectors of each image in the dataset and the vector of the target image. It also computes Sobel gradients and incorporates them into the similarity calculation for every image. DNN cosine similarity and Sobel similarity are used to create a weighted final similarity score.
The target image path, the CSV file path containing feature vectors, and the directory path holding photos are all specified by the user in the main function. Similar photos are retrieved by the dnn_image_retrieval function, which sorts them according to similarity scores.

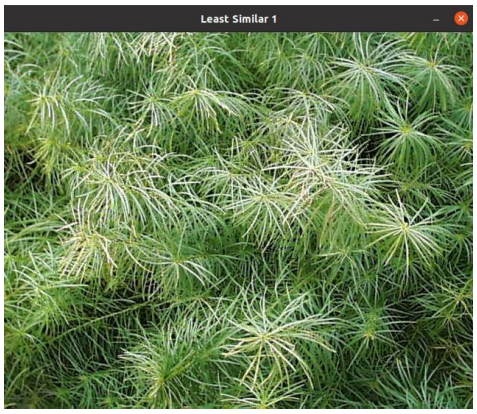
Target Image(Clock)

Similar Images





Least Similar Images



ACKNOWLEDGEMENTS:

I am grateful to the OpenCV documentation, which has been a vital resource during this project. Understanding and putting different computer vision algorithms and techniques into practice has been madepossible by the comprehensive explanations given in the OpenCV documentation.

I would also like to thank my peers for their help and feedback during this project. Throughout this process, exchanging knowledge, working together to troubleshoot challenges, and sharing ideas have all been extremely beneficial.