Introduction to Computer Vision (ECE 332)

Final Project Report

IMAGE GOOGLE

Name: Aakin Desai

Contents

1. Views about CV and its Applications 2

2. Project Description 3

3. Design, Approach and Implementation 4

4. Result 10

5. Analysis 10

6. Remarks and Future work 14

7. Feedback 15

Views on Computer Vision:

In layman language, computer vision can be understood as the ability of a computer to mimic the human visual perception system i.e. understanding images, retrieving information and recreating different structures from the images. The research which started way back in the mid-20th century has only seen more challenges in its journey. It all started with training robots to capture images and retrieve the information in the images, sooner digital image processing made its importance among scientists. Russell has made a very early contribution by inventing a digital image scanner which scans the image in binary form and inserts play enlarging the image using the binary format. it is paved the way for today's technology of digital image processing. another research that needs to speak addition is the attraction of 3D representations from 2-dimensional images. It is well known that the birth of computer vision happened at MIT when a group of researchers and MIT students tried solving the machine vision problem which is differentiating the background from the foreground and establishing the images which are not overlapping in the real world. However, this has been a challenge even today which implies that we are nowhere near solving the computer vision problem. there have been many research outcomes in this field, but nothing can completely solve computer vision. the techniques or phenomena that I have learned in this class. I have talked about the principles that I have understood and the challenges that I faced over the course of this class below.

starting with binary image processing there we tried to label the images connected component labeling in which he tried labeling grayscale images using the algorithm. We continued it by using morphological operators for enhancing the image. Later we used color histograms 2 correct images based on illumination and gaussian kernels. However, the tricky parts or the challenging

parts were edge detection, where we developed a canny edge detector to detect the edges inside an image. It is done using the principle developed by canny (low and high threshold combination of the edge). But this doesn't always produce highly accurate results as the input parameters depend on the type of the image etc. Later we talked about the implementation of texture detection using the structure of the parent block and blending multiple duplicate blends using image blending principles. One thing that I felt which is necessary is having the knowledge of mathematics which deals with linear algebra, similarity distances, probability, and statistics. Computer Vision is a great collaboration of physics, mathematics, and statistics powered together using computation. There are many applications in this field that start from the cameras that we use in our smartphones to satellite images of mars etc. the possibilities are endless in computer vision so is its completeness.

Applications:

Talking about the applications, when the term computer vision wasn't even coined, Hubel and Wiesel tried to understand the stimulus of cat' neurons to their perception of different images. This wasn't very successful though. But it paved the way for state-of-the-art computer vision applications over the period. Every sector like healthcare, automotive, agriculture, banking, industrial, retail and retail security, etc have their own applications of computer vision. We all know about self-driving cars that use object recognition, lane detection, motion detection, etc which fall under CV. It is one of the major applications in today's world. Even in healthcare, the possibilities are endless where we can try to detect the diseases from the history database and also using nano camera capsules to understand the functions happening inside the body. These

all use pattern recognition, object recognition, etc. and are highly computational and can be done using convolutional neural networks.

Amazon Go uses cameras installed above the item shelves which recognizes the items picked by the customer and adds it into his cart and bills it automatically. There are flaws in such systems, but it only shows a way forward to improve the research capabilities. In agriculture or livestock farming, we can implement CV to monitor the animals, plant life, height, response to different conditions, etc. this helps to keep in track the real-time response. In controlling traffic accidents etc. the police department uses visual tracking of the number of plates and sends tickets to them directly in real-time. Gaming consoles that use VR AR and gesture recognitions are also some great applications. Samsung has introduced gesture control on their smartphones which also has OCR on their gear watches. Gauss has been developing CV systems to detect skin related diseases like psoriasis and its stages. The applications are endless and possibilities for contribution are also endless.

Project Description:

As a part of our course, we have decided to build an image search engine for our final project. Unlike normal word search engines, we use reverse image search where the query is the image and we search for similar images in the dataset. The term similar here means a lot of things. For instance, it can be a similar color, structure, objects, faces in it, etc. Why is this engine useful? Suppose we came back from a vacation and wanted to sort out the images and delete the useless pictures or edit the ones which look similar. It is a tiresome process to sit down and sort

thousands of images on our cameras and smartphones. This engine helps you to sort out such images and store them in clusters.

To successfully build this project we had to break it down into four major milestones. Of all the milestones the most important is the one where we find the similarities using different image descriptors and store their features.

Design and Approach:

As we currently do not have a large dataset, we have utilized the holiday dataset of INRIA in association with the Advestigo travel agency. They have over 1500 images of different locations and versatile images of different objects. Technically, these search engines are called Content-Based Image Retrieval systems.

The initial process for such engines is to pre-organize our dataset to decrease the computational effort. This can be done by indexing the dataset with allocating features for each image and storing them for later purposes. This reduces a lot of computational time as we must deal with the feature's files only and not the images for every single search. These features are the outcomes of the image descriptor that we use. Image descriptors are something which defines the contents inside the image, they can be the color histograms, gradient's magnitude and direction, scale-invariant features, segmentation and region adjacency graphs. There are many descriptors to use and each descriptor has a different feature outcome.

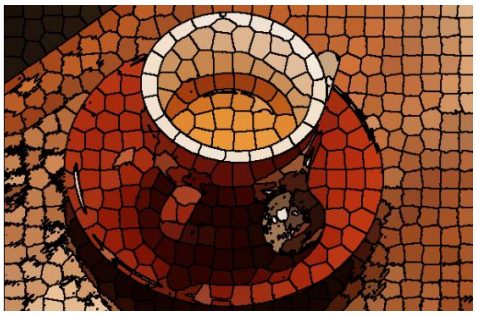
For the color histograms, the features are the quantized values of the bins of the color space that we use (can be HSV, RGB, etc.). If we use the SIFT descriptors, we can store the feature points as the feature vectors and later use them comparison. In RAG based descriptors, we can store the

edge-weighted values as features. The next step is to compare the features obtained from the descriptors that we used. We can simply compare using Euclidean distance of the color histogram. There are other similar functions that we can use to compare the features of the database with our query.

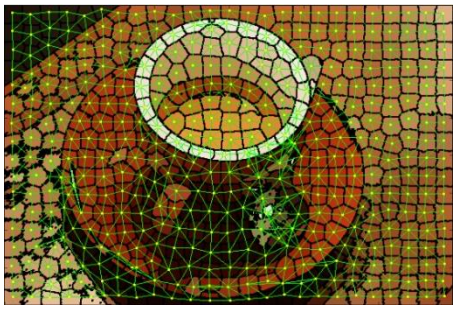
I took the part of programing the image descriptor for RAG based. Then, I implemented the Hausdorff's distance to compare the features of the dataset and the query.

RAG (Region Adjacency Graph):

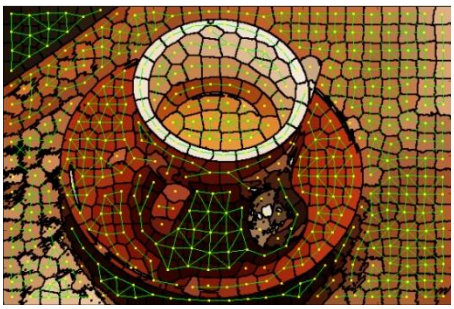
Region Adjacency Graphs, as the name, proposes to speak to the nearness of areas with a chart. Every region in the picture is a hub in a chart. There is an edge between each pair of the nearby regions (districts whose pixels are adjoining). The weight of between every two hubs can be characterized in an assortment of ways. For this model, we will utilize the distinction of normal shading between two areas as their edge weight. The more comparable the areas, the lesser the weight between them.



a)Segmented Regions



b)Nodes and Edges of the RAG

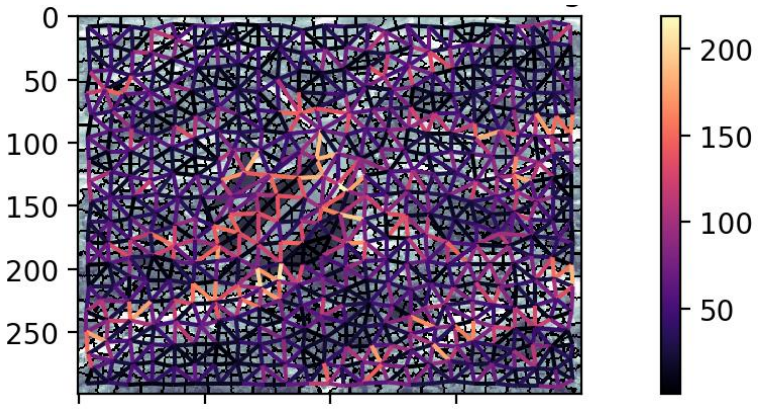


c) Components divided

This is a structure-based algorithm where we segment the image into fine regions using K-Means clustering and then talk about the regions on either side of an edge (adjacent regions). This method is generally used for image segmentation where we segment the image into different regions by comparing the adjacent regions of an edge. This can be seen in the following pictures.

Before segmenting the image, we use K-means to divide the image into regions of the same mean color. (i.e. we take a cluster of pixels in the image and compute the mean color and allot the region that means color). We can see that the first picture below represents the regions based on the mean color.

Each region depicted is a node in the RAG. Two adjacent nodes are connected by an edge as shown. Hence, every edge represents the two nodes that it connects i.e. the two regions are connected by an edge. Each edge represents the weights of the connected regions. It can be anything like the relation of mean color or size of the region etc. for segmentation purposes if we consider the mean color of the regions of one edge we can decide if we want that edge to be present or not. If the mean color is close, then the weight is less. Basically, weight is a parameter to judge if two regions are similar or not. In the third figure, we can see that similar color regions are connected, and all the different regions are distinguishable.

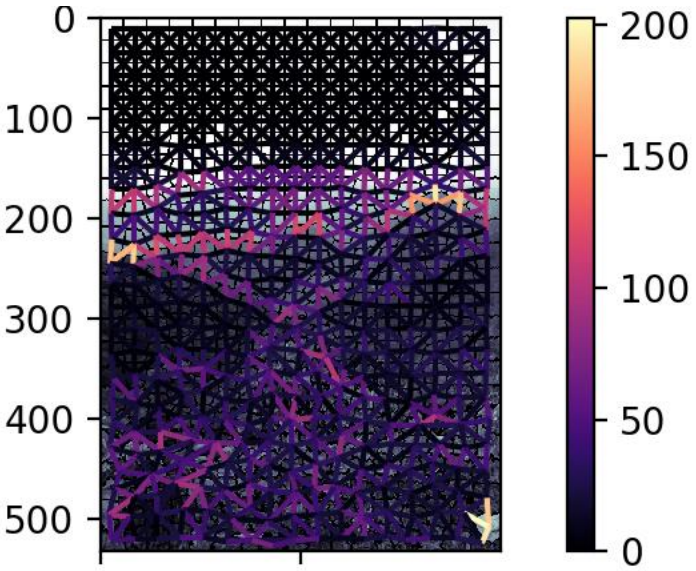


a) Image in the dataset

b) RAG of the image shown on left

The similarity index to form independent regions can be given as a threshold so that we can decrease or increase the area of the formed segments in the image.

How can we apply this method to compare two different images? Well, we need to compute the RAGS for each image in the dataset and store the nodes and edges as the features of the image. (In the case of the color histogram, the features were histogram quantization values of each bin) The difficulty with this that the computational effort for extracting RAG features from one image takes at least 3 seconds of time. But once we compute and store the RAGs of all the images in the dataset it is easy to compare the query with the features directly. The next question is how we compare the features. This can be done using the Hausdorff's distance between the weights of the edges of RAG of the images.



a) Image in the dataset

b) RAG of the image shown on left

Hausdorff's distance is a measure of the distance of two subsets in a metric space. If there are two subsets X and Y in a space, then the Hausdorff's distance is

$$d_{\mathrm{H}}(X,Y) = \max\left\{ \sup_{x\in X} \inf_{y\in Y} d(x,y), \sup_{y\in Y} \inf_{x\in X} d(x,y) \right\}$$

[$G^d = (V^d, E^d)$ where V^d is the nodes of the segmented image RAG and E^d is the edges of the RAGs.]

In the above formula, we can see that we are inputting the edge weights of Gd (desired image) and Gm (model or database image). v^m and v^d are the weights of the edges. The weights we chose here is the mean color of the regions connected by the edge.

Now, we use the inbuilt function for Hausdorff's distance in python for calculating the difference between the features of the query image and the image from the dataset. Now, our features are the weights of the edges of the RAGs of the images. However, the images may not be of the same shape and size. We need to scale our images into a specific size and then compute the distance. We need to input the weight arrays into the Hausdorff's function, it then estimates how close the weight sets are by seeing how close each point in one set is with some point in the other set. But we need to take the maximum of the symmetric distance i.e. to calculate the distance from both sides and consider the maximum.

Now, after computing this distance, it is now time to show the results by using the index of the feature of the resulted images in the dataset. We can check the precision by comparing the original results with the results obtained from this method.

Result and Analysis

Global Color Histogram result: -

In Global Color Histogram, we are calculating the number of pixels for each of the bins. After that we are normalizing it, so size of image does not affect result. Most important thing in this project is finding a way to compare an image from dataset to a query image. We used Chi-distance to calculate the distance between vectors (Normalized Number of Bins) of two images. After that we tried two method to get our results.

- Setting Threshold Value: - In this we set 1.75 as threshold value and found all images that satisfy this condition
- Top Images: - Secondly, we sorted an image on basis of its distance and took top 5 images as our results.

Query Image: -

Result Image using Setting Threshold Value Method: -

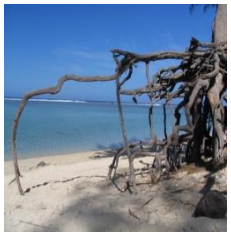







Result Image using Top 5 Images: -





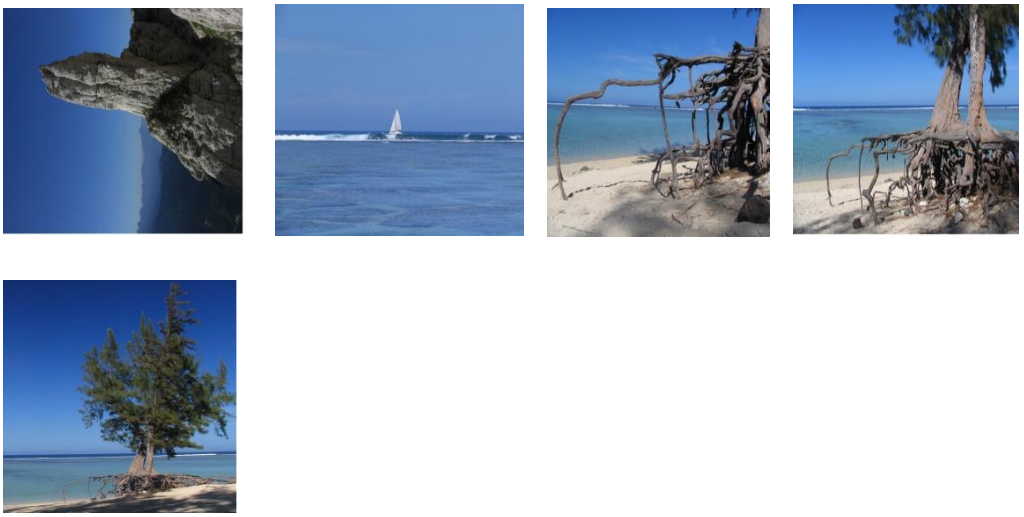
Regional Histogram result: -

In Regional Histogram, we are dividing an image in five rectangles with center rectangle covering a maximum area. As it contains more useful information about an images or in other words we can say that it contain most important information about an image, Then we will follow same step as in Global Color Histogram only difference is that number of vector for each image is now five times that of Global Color Histogram. Therefore, it should better results as compared to Global Color Histogram.

- Setting Threshold Value: - In this we set 11.75 as threshold value and found all images that satisfy this condition
- Top Images: - Secondly, we sorted an image on basis of its distance and took top 5 images as our results.

Result Image using Setting Threshold Value Method: -

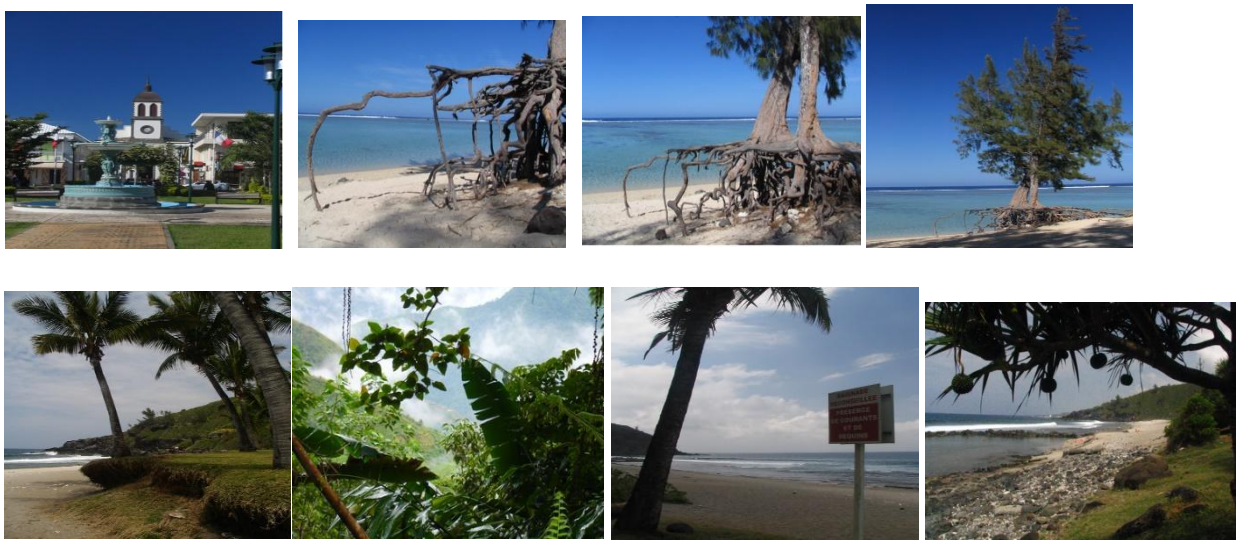
Result Image using Top 5 Images: -



3. RAG based Description is method which a given image is divided in regions of similar features such as on basis of color intensity, key point descriptor and many more. We took weight of a region as method to compare a query with data set image. Weight of a region is an index which used to compared similarity of an image with respect to neighboring regions. Unlike a color histogram method, it is used to compare the features rather than only color. But results were not so good in our case. Reasons we found that there is possibility of similar features in image but of very different type. For example, there could be chair at beach as well as at restaurant. So, it will match our image but in actual it is not so. To confirm our results, we compared results of each method using similarity index method and as well as ORB method other than precision recall method. Similarity index method confirmed us that result of our RAG

based method similarity index came to be large as compared to other methods. At the same time this method is computationally to robust. As it took us around 3 seconds per each data image. Our data set had 1500 images it took us around 4500 seconds to run it.

Result Image using Setting Threshold Value Method: -



Result Image using Top 5 Images: -

Comparison: -

Precision and recall: Precision is the ratio of correct image among the retrieved instances to total number of images retrieved. while recall is the ratio of the total amount of relevant images that were retrieved.

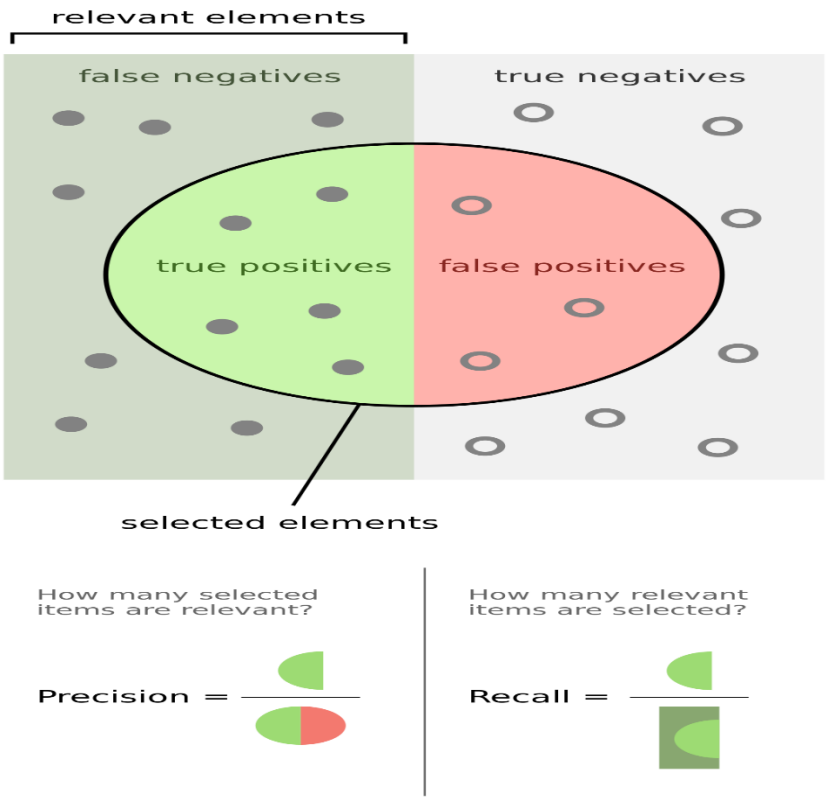


Fig: "https://en.wikipedia.org/wiki/Precision_and_recall#/media/File:Precisionrecall.svg"

Both factors are combined in the F1-score.

$$F1 = 2\frac{P * R}{P + R}$$

Structural similarity index is index which is used to compare structure of two images. It values lie between zero to one.

Table: -

Method	P	R	F1-score	SSIM
Global Color Histogram	5/7=0.714	5/10=0.5	0.588	0.53
Regional Color Histogram	8/12=0.67	8/10=0.8	0.729	0.57
RAG Based Description	3/8=0.375	3/10=0.3	0.33	0.63

Future Work:

The image search we built is a basic version of what google or Tineye has to offer. They operate on over 2 billion images. Also, the accuracy of their systems is higher. There are many reasons for that. For example, Google uses metadata too along with their advanced mathematical algorithms in the description of their database.

Tineye, on the other hand, can also extract the similar images for the edited or filtered query images. This is something that can be implemented if we had more time. Since people use a lot of filters these days on enhancing their photographs, we might miss out our original raw photos. This can help to find them.

Other algorithms to improve the efficiency are using the SIFT algorithm, Maximally stable extremal regions, Vocabulary tree algorithms.

We can also implement a hybrid search engine by using the metadata (tagging) i.e. we can tag the objects inside the image as the depiction of the image and search for objects plus image as the query.

Course Feedback and Suggestions:

The structure of the course was crisp and well-thought. It was planned in an orderly manner starting right with the basics and testing those with proper machine problems every week.

I feel that those machine problems did really help a lot in understanding the concepts with clarity and applying them in our programs.

It is not only important to know the algorithms but, it is as well important to apply them for real time use. So, this course was the right platform for beginners in computer vision to learn about the basics rather than directly using the libraries available for programming purposes.

As the world is moving into deep learning extensively, professor rightly mentions about the importance of CV and how deep learning can be used as a tool to solve problems rather than saying deep learning itself is computer vision.