Gautam Jonnalgedda

Niharika Nishin Patil

COEN 345

Computer Vision II

Image Search Engine

1. Introduction

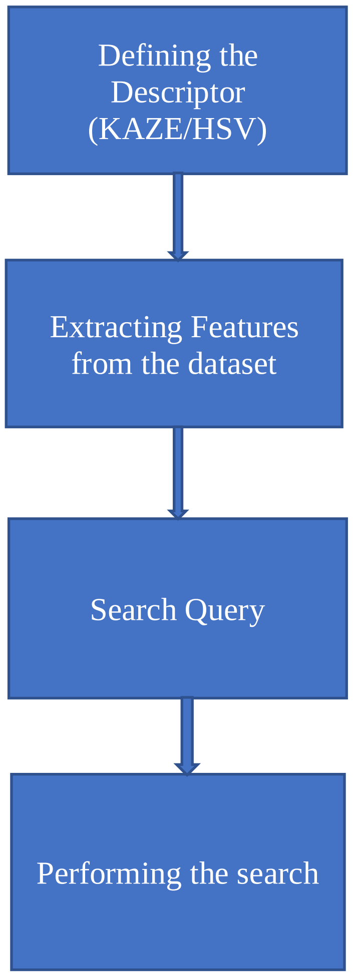
Text based image search engines like Google Images, Bing, Flickr are based on key word searches based on tags. This is a great method but requires manually tagging & annotation of images or is dependent on contextual hints such as the text on that image on a webpage. This is a time-consuming process and is based on giving an 'old school' query to the engine which gives the images with the similar tags back as a result. The proposed implementation uses image as an input to the engine and this gives us similar results. The approach implemented is heavily derived from [1]Adrian Rosebrock's implementation of his Image Search Engine. This is based on two image descriptors: one being an HSV descriptor and the other based on the KAZE descriptor.

We have used a dataset consisting of predominantly vacation photos from different places of the world and other random photos as would exist in the database of Google or Bing and then used this approach by giving an input image to our engine which returned 15 images to us. Some which consisted of values similar to our search query and some other images. The dataset is a taken from NRIA Holidays Dataset quoted by Rosebrock it is a 'search by example' method.

2. Methodology & Results

Two descriptors are used in this implementation namely: HSV color descriptor & KAZE descriptor.

Flow:


Defining the Descriptor (KAZE/HSV)
Extracting Features from the dataset
Search Query
Performing the search

2.1. HSV Color Descriptor

Algorithm used for this purpose:

Step 1: Using a 3D color histogram as our Image Descriptor
Step 2: Extracting Features from the dataset and storing them
Step 3: Developing a search
Step 4: Executing the search

HSV descriptor is a 3D color descriptor which is conceptually a 3D histogram in the HSV color space (Hue, Saturation and Value).

The color wheel of the HSV is portrayed as a cone or cylinder but always with these three parts:

Hue

Hue is the color part of the model, represented as a number from 0 to 360 degrees: As follows:

Color	Angle
Red	0–60
Yellow	60--20
Green	120–180
Cyan	180–240
Blue	240–300
Magenta	300–360

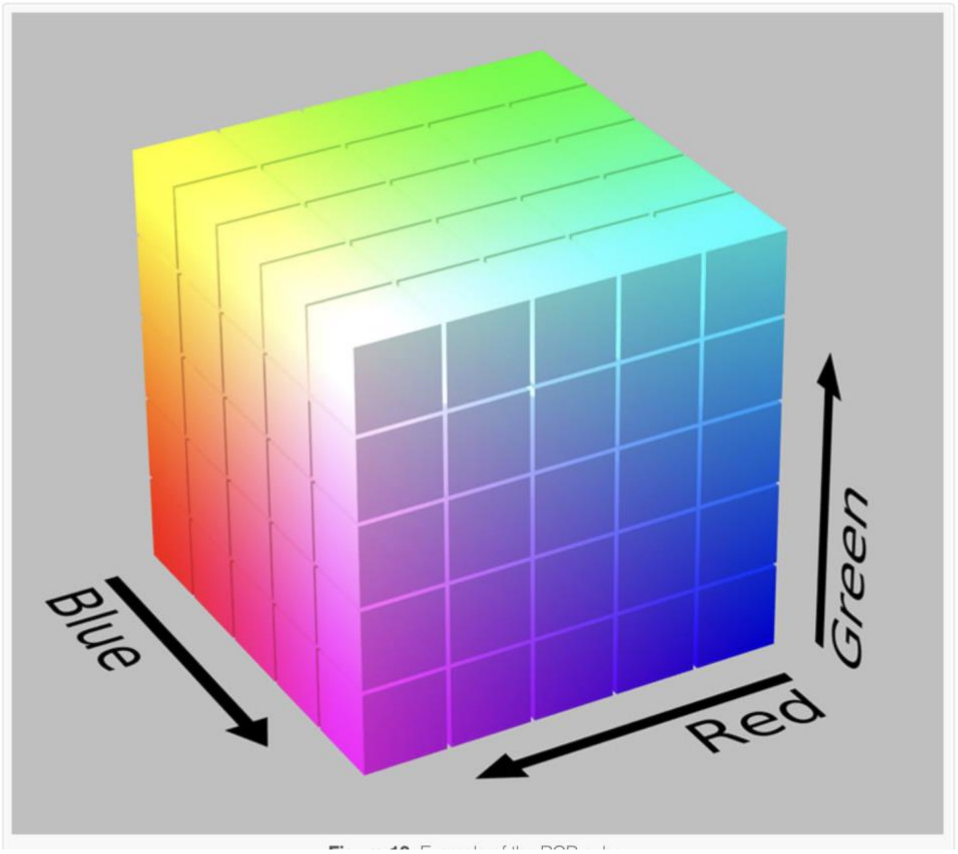
Saturation

Saturation is the amount of gray in the color, from 0 to 100 percent. Decreasing the saturation toward zero to introduce more gray produces a faded effect. Saturation is expressed in a range from just 0-1, where 0 is gray and 1 is a primary color.

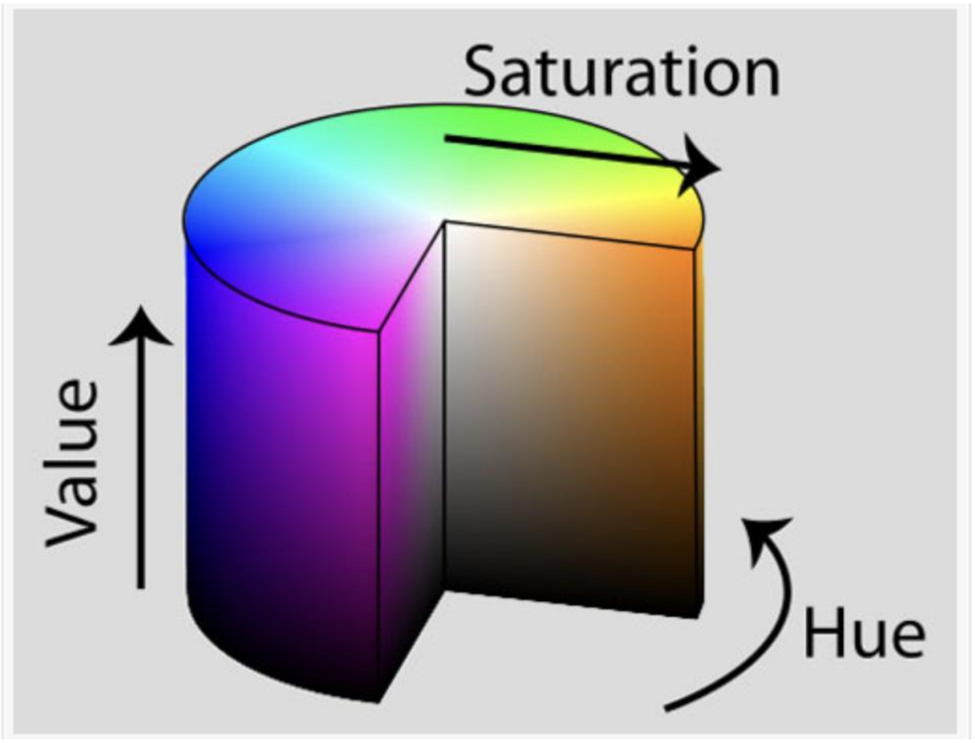
Value (or Brightness)

Value represents the intensity of light or brightness where 0 is black and 100 is the brightest and results in the max color.

We convert our RGB image into HSV model:

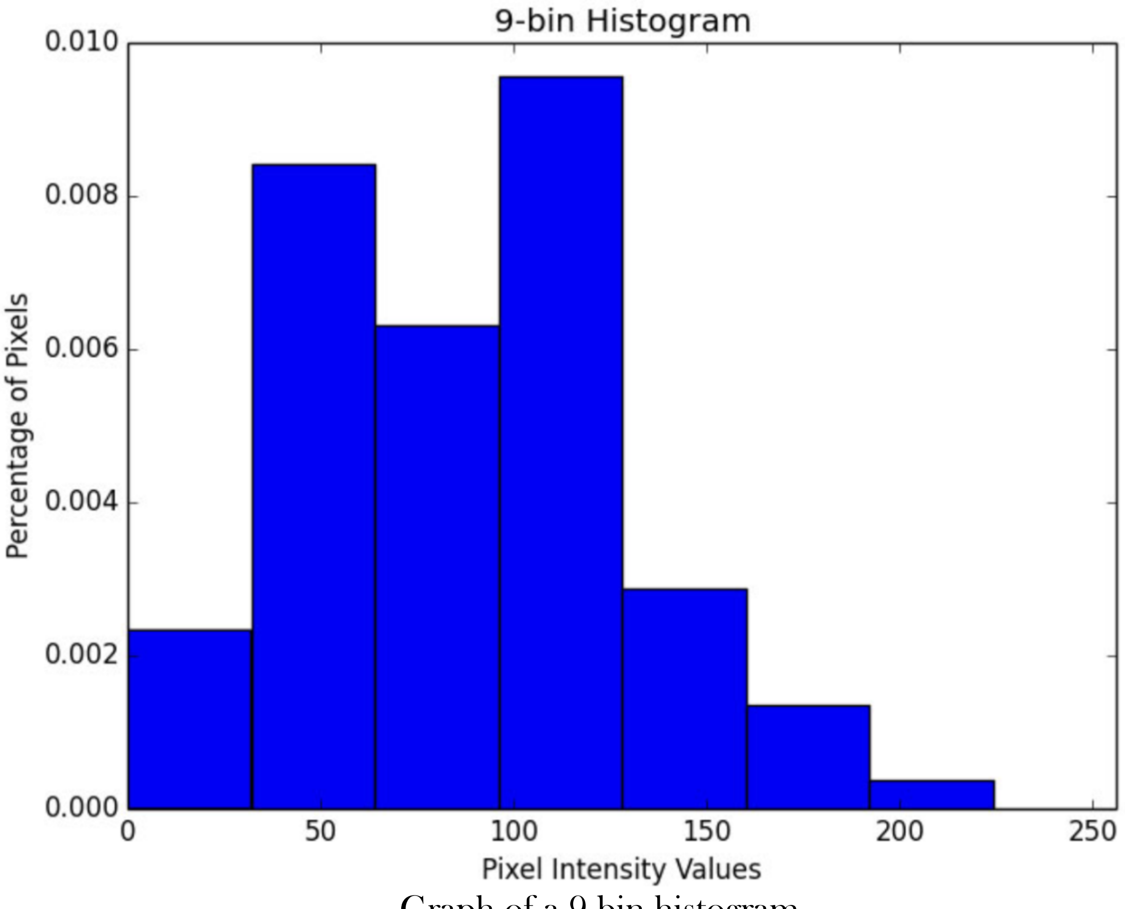


RGB Model

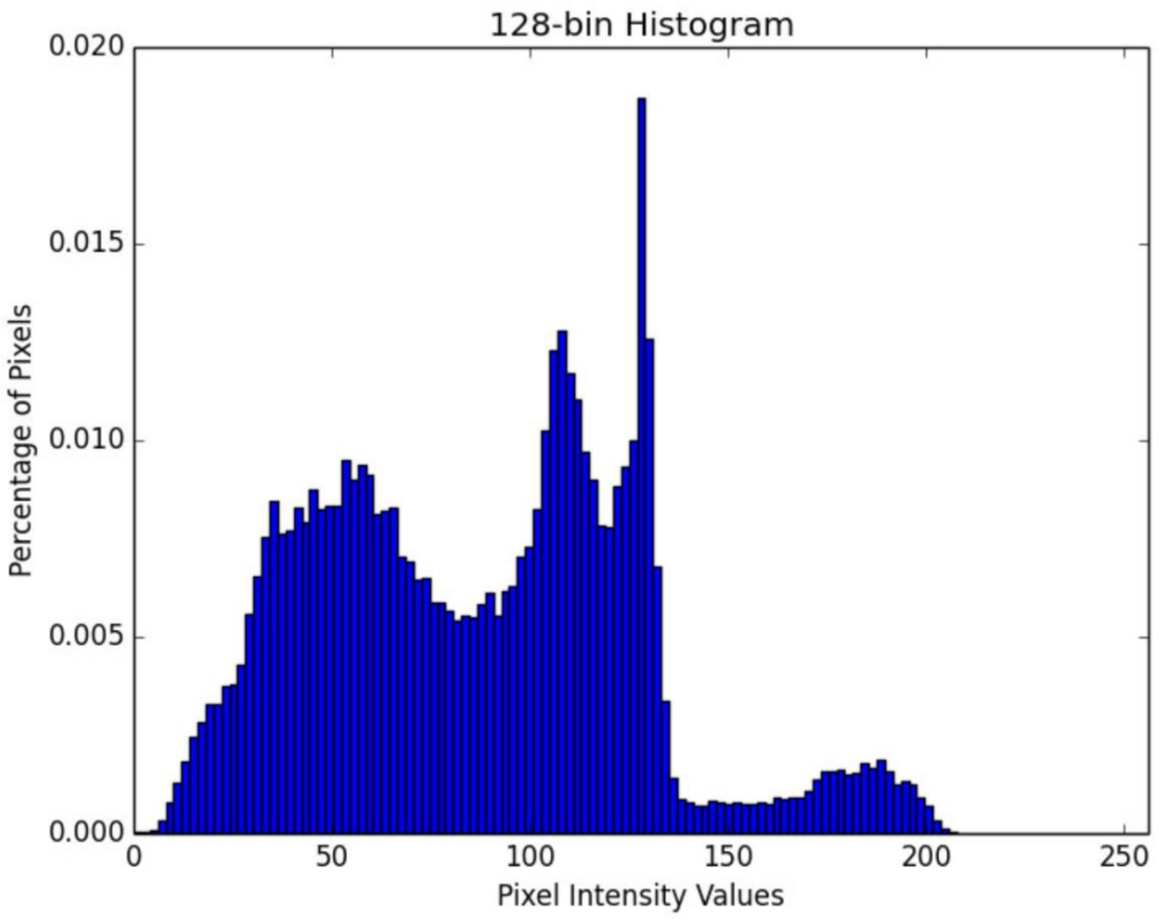


HSV Model

Histograms are used to represent the density of pixel intensities in an image pertaining to a specific color. The histogram used here will give the probability density of the underlying function, or in this case, the probability **P** of a pixel color **C** occurring the input image.
There is a trade-off with the number of bins selected for the histogram. If *too few bins* are selected then the histogram will have less components and won't be able to distinguish between images with varying color distributions. If *too many bins* are used, the histogram will have too many components and similar images may be regarded as not similar when in they actually are.



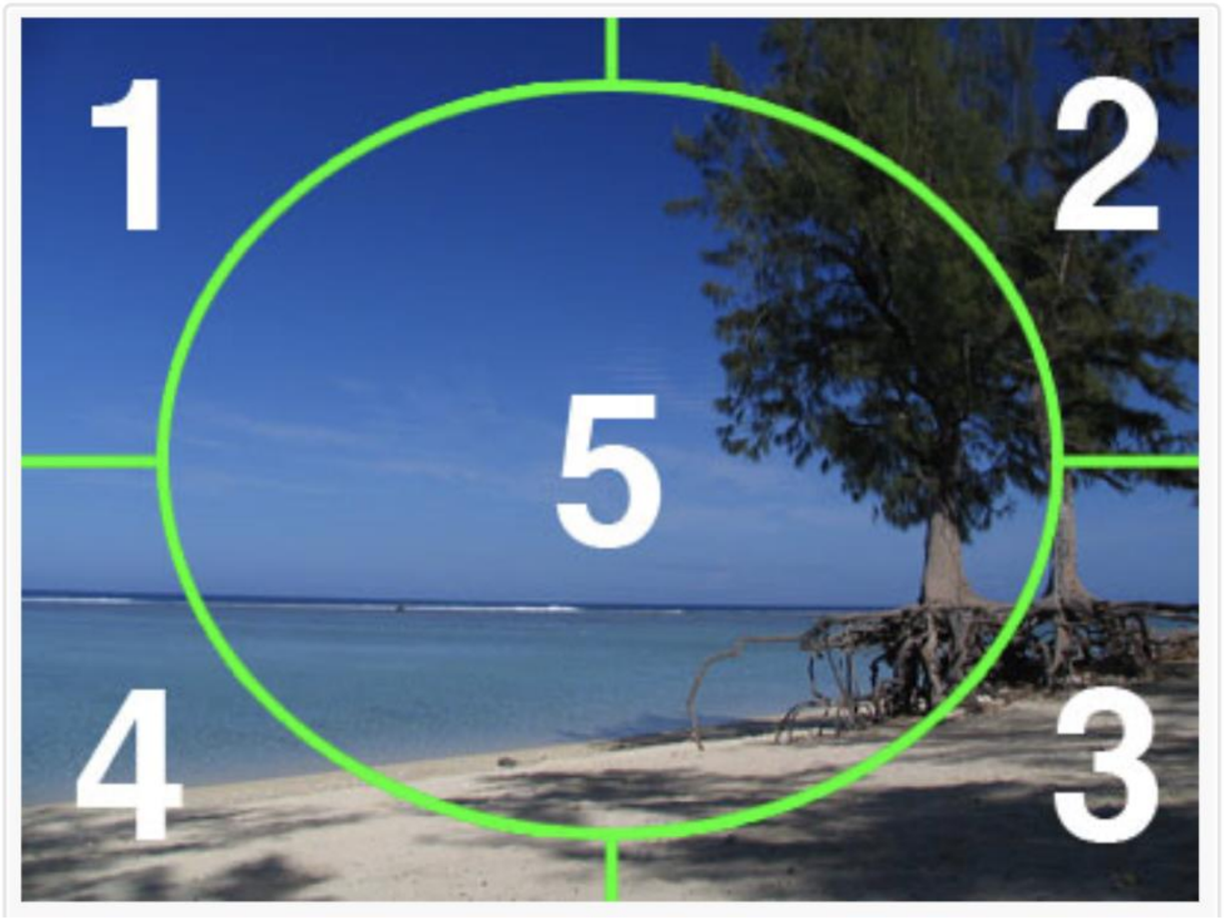
Graph of a 9 bin histogram



128 Bin Histogram graph

For our approach we are using HSV color space with 8 bins for the Hue channel, 12 bins for the saturation channel, and 3 bins for the value channel, yielding a total feature vector of dimension *8 x 12 x 3 = 288.*

For our HSV descriptor we divide the image into 4 segments i.e. Center, top-left, bottom-left,top-right, bottom-right. This forms the basis for our region based color-descriptor.



Descriptor after breaking the image into 4 parts

We calculate the histogram for each of these segments and make up our feature vector.

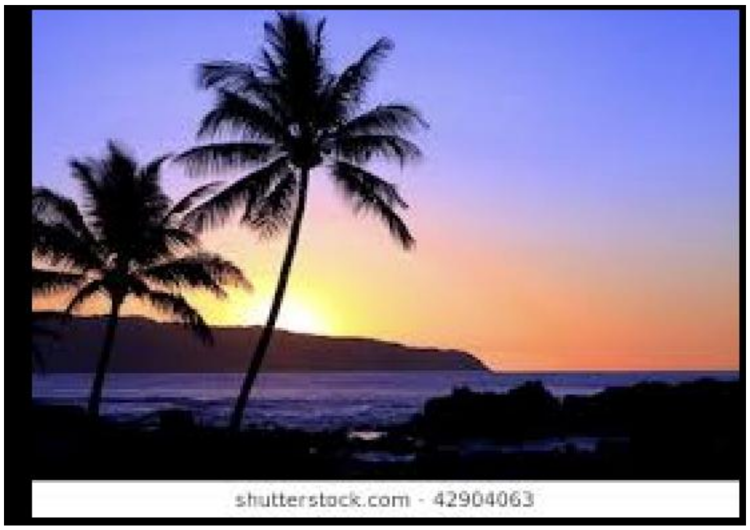
Now our feature vector contains 1440 features.

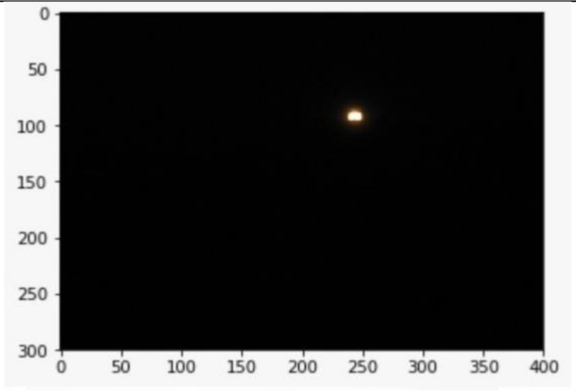
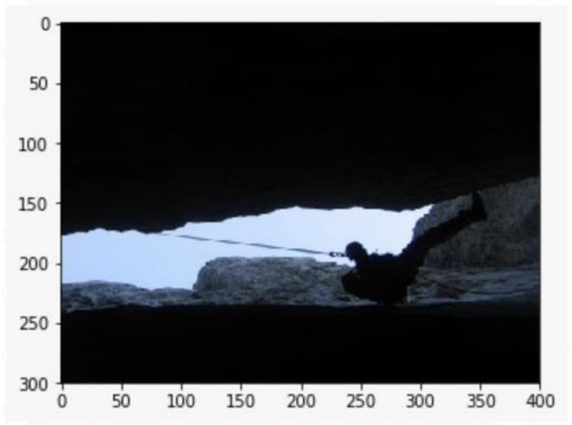
As 288x5=1440

We will now use similarity metrics to check which images are most similar. We will use chi- square & cosine-distance to get the results.

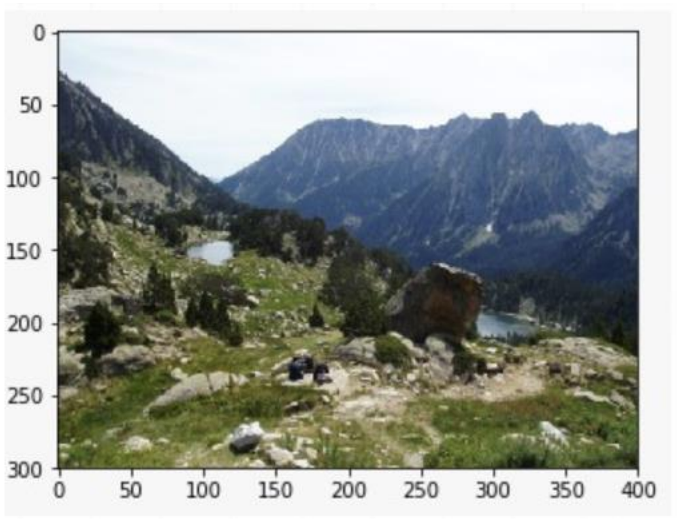
Results:

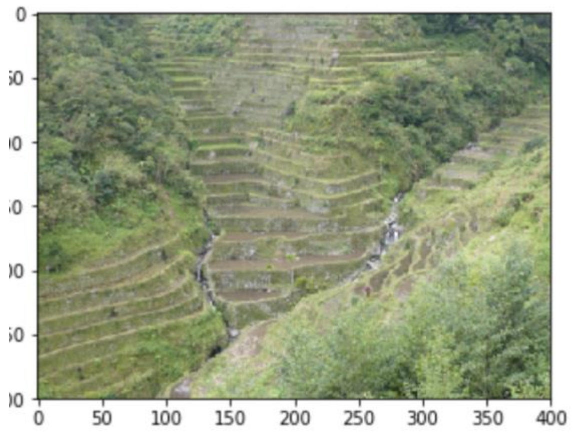
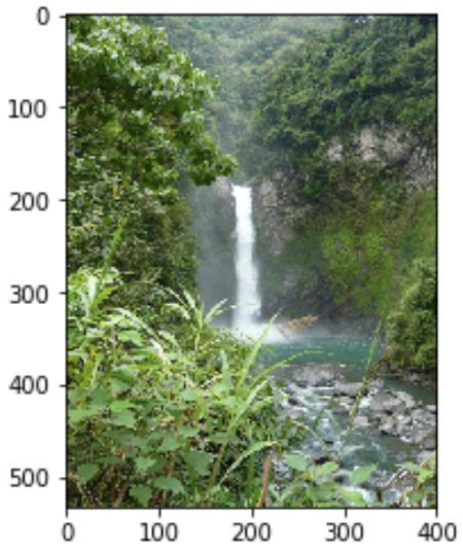
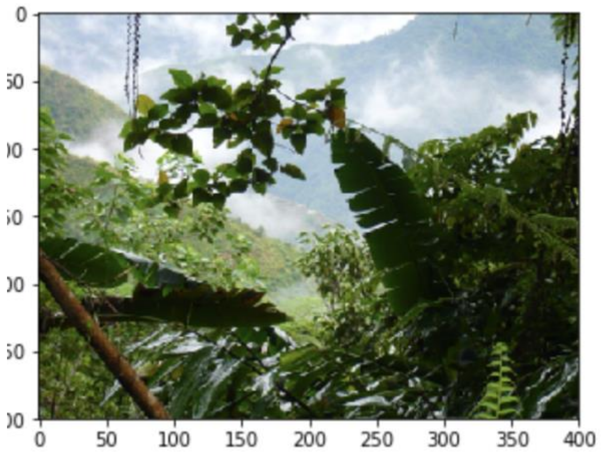
Query Image:



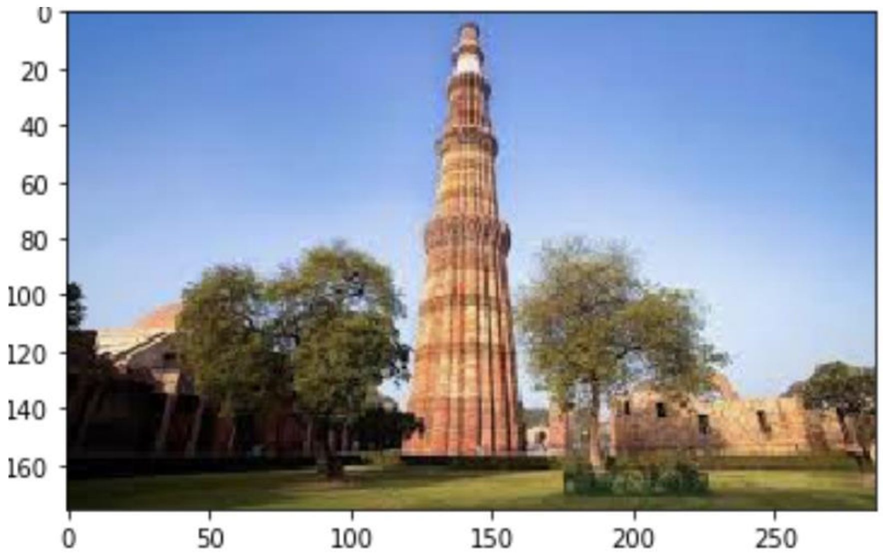
Results on using Chi Square	Results on using Cosine Distance
	
	

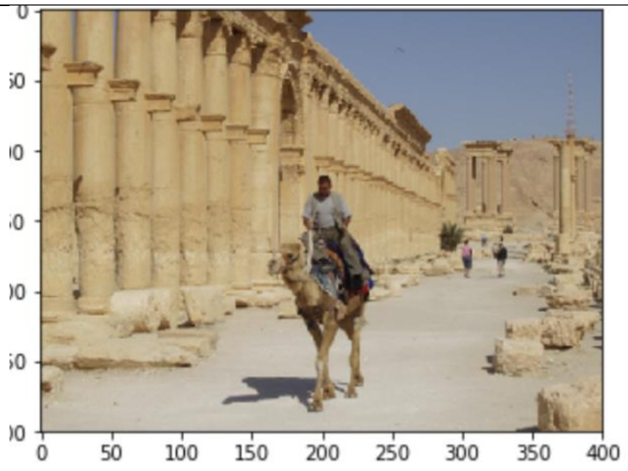
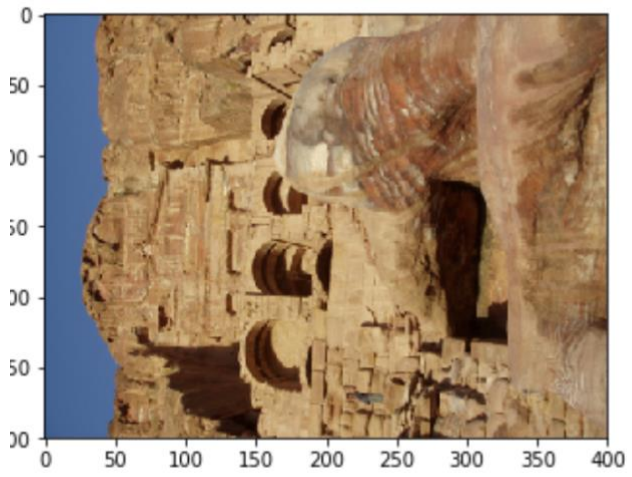
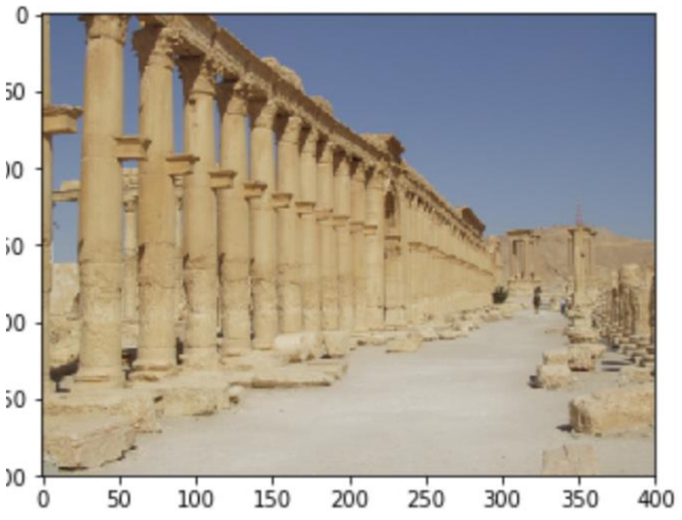
Query Image:



Results on using Chi Square




Query Image:



Results on using cosine distance




2.1. KAZE Descriptor

KAZE is a Japanese word that means wind. Wind is the flow of air on a large scale who's flow is ruled by nonlinear processes.

[2]KAZE features is a novel multiscale 2D feature detection and description algorithm implemented in nonlinear scale spaces. Other methods detect and describe features at different scale levels by building or approximating the Gaussian scale space of an image i.e DOG images. But Gaussian blurring does not regard the natural boundaries of objects and smoothens to the same degree both details and noise, reducing localization accuracy and distinctiveness. In the KAZE implementation detection and description of 2D features takes place in a nonlinear scale space by means of nonlinear diffusion filtering. Thus blurring is made locally adaptive to the image data, reducing noise while retaining object boundaries, obtaining superior localization accuracy and distinctiveness. The nonlinear scale space is built using efficient Additive Operator Splitting (AOS) techniques and variable conductance diffusion.

Nonlinear diffusion methods perform much better than linear ones [5,6] and give more accurate results.

Nonlinear diffusion approaches depict the luminance of an image through increasing scale levels as the divergence of a certain flow function that controls the diffusion process. Nonlinear partial differential equations (PDEs) are used due to the nonlinear nature of the involved differential equations that diffuse the luminance of the image through the nonlinear scale space. The below eqn shows the classic nonlinear diffusion formulation:

$$\partial L\ \partial t = \mathrm{div}\,(c\,(x, y, t) \cdot \nabla L)$$

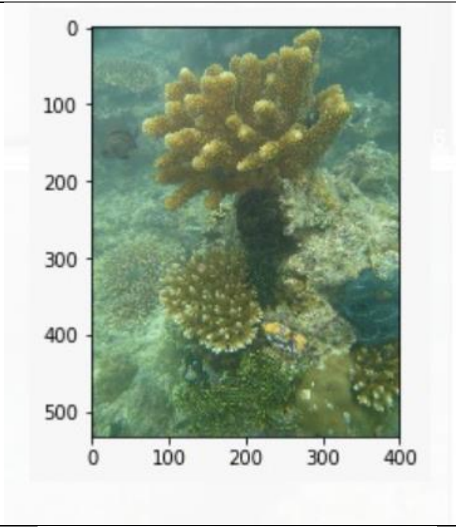
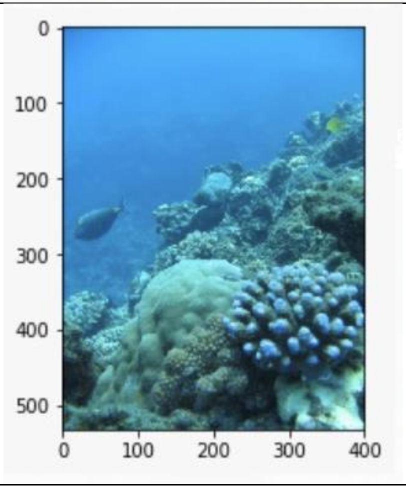
where div and ∇ are respectively the divergence and gradient operators. The conductivity function (c) in the diffusion equation, it is possible to make the diffusion adaptive to the local image structure. The function c depends on the local image differential structure, and this function can be either a scalar or a tensor. The time t is the scale parameter, and larger values lead to simpler image representations. The Kaze implementation used by us is based on Andrey Nikishaev.

The feature vector descriptors is derived from the keypoints, each descriptor has size 64 and we use 32 such and thus the feature vector has a dim of 2048.

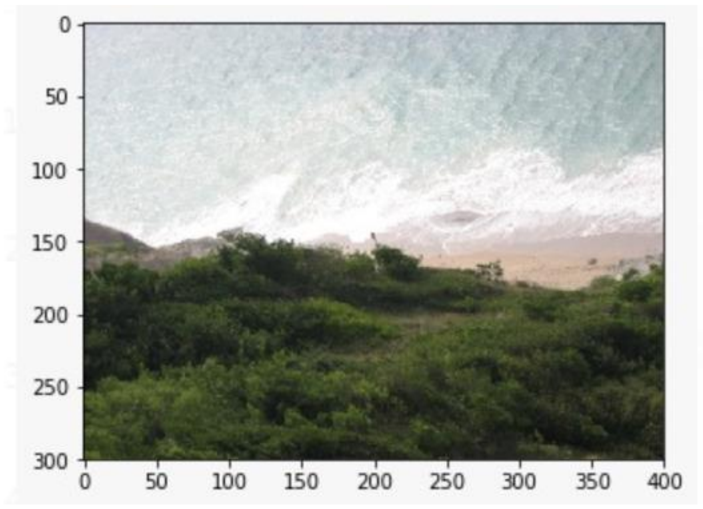
We then match the features of the query image and the images in feature dataset using cosine distance and return the top N features.

Results:

Query Image:

Results on using Cosine Distance



Query Image:



Results on using Cosine Distance

3. Observations:

1. Kaze descriptor works better for natural images as compared to Histogram.
2. Between chi square & cosine distance, to the naked eye cosine distance seems to deliver better results.

4. Possible Improvisations:

We can improvise our results in the future by doing the following:

1. One can use more distance measures such as Bhattacharya distance, Intersection, Correlation coefficient.
2. Using other descriptors like SIFT, SURF & ORB.
3. Using a bigger dataset and compute cosine distance efficiently using Annoy Index (which speeds up cosine distance considerably). 1 million images can be searched in just 2ms.
4. Evaluation Metrics to understand which algorithm works better.

5. References & Literature Review:

[1] https://www.pyimagesearch.com/2014/12/01/complete-guide-building-image-search-engine-python-opencv/

[2] https://www.doc.ic.ac.uk/~ajd/Publications/alcantarilla_etal_eccv2012.pdf KAZE Features. By Pablo Fern´andez Alcantarilla1, Adrien Bartoli1, and Andrew J. Davison

[3] https://medium.com/machine-learning-world/feature-extraction-and-similar-image-search-with-opencv-for-newbies-3c59796bf774

[4]https://www.kaggle.com/jessicali9530/stl10

[5] Weickert, J., ter Haar Romeny, B., Viergever, M.A.: Efficient and reliable schemes for nonlinear diffusion filtering. IEEE Trans. Image Processing 7 (1998)

[6] Ter Haar Romeny, B.M.: Front-End Vision and Multi-Scale Image Analysis. Multi-Scale Computer Vision Theory and Applications, written in Mathematica. Kluwer Academic Publishers (2003)

[7] http://lear.inrialpes.fr/people/jegou/data.php