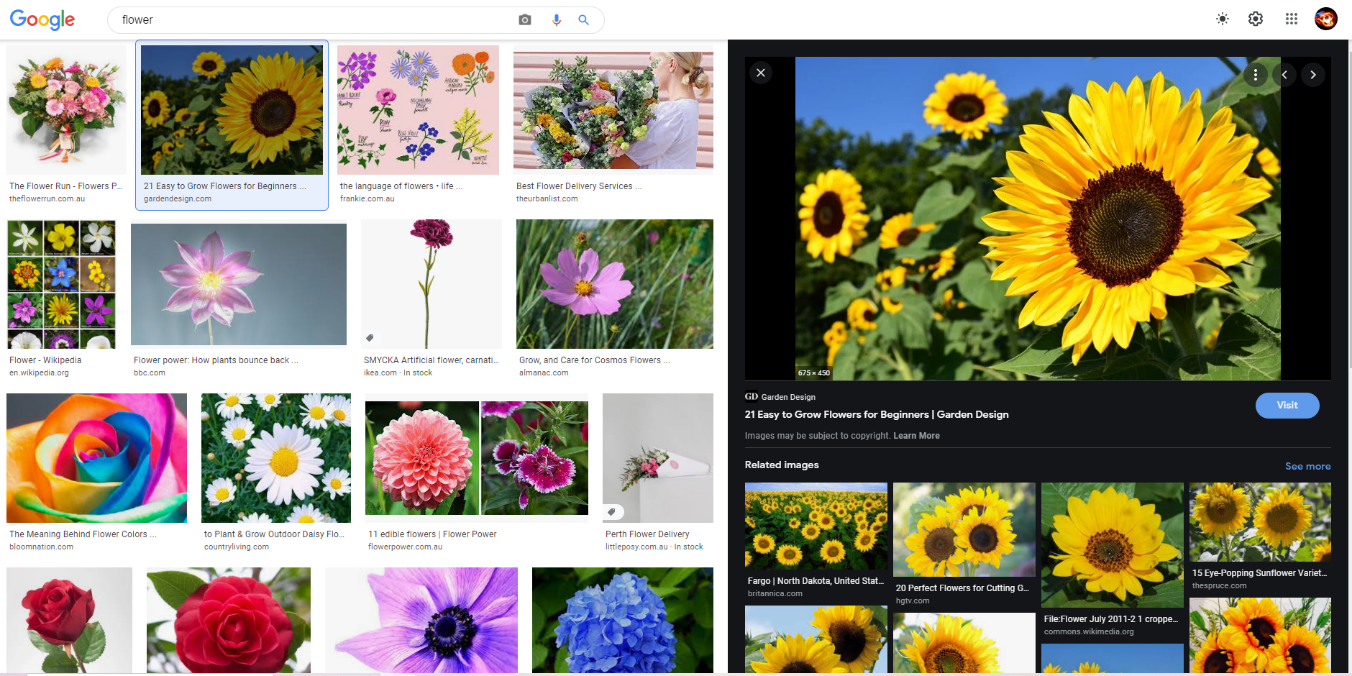
Image Based Retrieval Project

By: Sean Oldenburger Date: 09/07/2021

**Introduction:**

An image retrieval system is a program used for browsing and searching images from a large database, such as google. The system looks at the image you have selected, the query image, and ranks all the images in the database from most to least similar and presents the top k number of similar images to you.

In a real-world example, if you google “flower” and look at the images, clicking on the sunflower (the 2nd image) there will be multiple images under the opened image. As you can see, these images are very similar to the query/selected image. My goal in this project is to be able to replicate this system, however on a slightly lower scale as the google database is incredibly large.



*Figure 1: Example of googles Image Based Retrieval System on a sunflower.*

Throughout this project, I will be starting at a simpler version of the image retrieval system, comparing images together based on the raw pixel values, on low scale images. I will then move on to higher scale Images and start implementing feature extraction techniques on the images. I will be firstly exploring local feature extraction using Scale Invariant Feature Transform (SIFT) combined with the Bag-of-Words method. Lastly, I will be using deep learning algorithms for feature extraction, using Convolutional Neural Networks (CNN).

However, before I got into feature extraction and using different techniques, I needed to gain an understanding of how to measure success of the system and therefore will research metric techniques used in image retrieval systems. The two main metrics I will be using are called precision at k and mean average precision (mAP).

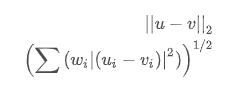
But first, I had to research a bit into what an image retrieval system is and how it calculates the distance between images to be able to find which is the closest or furthest image from a query/given image.

**Distance:**

As a first step I took the pixel values the images from the MPEG7 dataset which consists of 1,400 images of size 32 by 32 pixels. So being small scale images it was a good place to start. By converting every image of 32 by 32 pixels into a single vector of 1024 length (32\*32), we could use these vectors as our input data into the system. This means for every object (image) we had 1024 features, as each pixel value is stored as a feature.

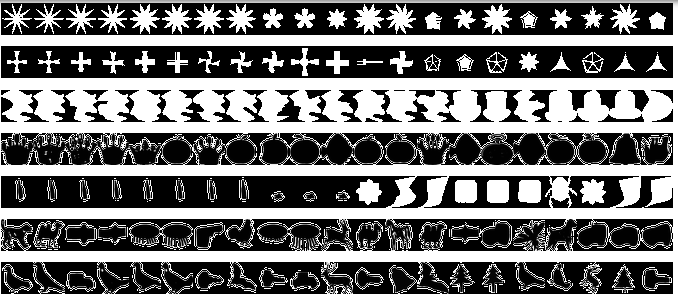
Second, we plot each object (image) on a 1024-dimensional graph (in theory as we cannot visualize this). From this we can find the distances between points using distance measurements such as: Euclidean distance, Manhattan distance, Canberra, Bray Curtis, or Cosine. All these techniques are similar and therefore for my project I have chosen to use Euclidean distance.

Euclidean distance computes the distance between two 1-dimension arrays (hence the reason we must reshape the object to be a vector). To compute the distance between 2 arrays we use the following formula:



*Figure 2: Euclidean Distance Formula.*

Using this formula, we can find the distance from a given/query image and find the distance to every other image (which is represented as a vector). We then sort out the distances from closest to furthest away and can display the top k amount, in the following example I have displayed the top 20 closest images based on the first image being the query/selected image.



*Figure 3: The top 20 closest images to the first image (query image) using pixel values as the features of each object and computing the distance using Euclidean distance.*

As you can see, these are very low scale simple images being only black and white and having very similar shape, rotation, and scale and therefore the image retrieval system performed decently, being able to display multiple correct/similar images in the closest 20 found. However, using pixel values as the features is not that effective as there are still a lot of incorrect images displayed.

**Metrics:**

Working onwards from the distance measurements we cannot manually check every single image query and decide to call the system a success or failure by viewing it ourselves. Therefore, we can evaluate some metrics which will provide each query with a value of success and give our entire system a single integer value to determine its performance.

For each query image we compute precision and recall.

* *Precision measures*: “out of all the images we retrieved as relevant how many are actually relevant?”
* *Recall measures:* “out of all the actual relevant images how many did we retrieve as relevant?”

For example, suppose a given query has 50 similar images to it, and the system retrieves 30 images out of a database that has 500 images. Out of the 30 images it retrieves only 20 were similar images. Therefore:

Precision = 20 / 30 = 66.7% Recall = 20 / 50 = 40%

However, the precision would be the same if the first 10 images were shown to be correct and the last 20 are wrong, compared to a query who shows 20 wrong first and the last 10 are correct. This is not ideal as an image retrieval system should favor having more correct images closer to the query. To allow for this we use a metric called average precision.

Query Image



Precision: 1/1 2/2 3/3 4/4 5/5 6/6 7/7 8/8 9/10 10/11 11/16

Precision: 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.9 0.9091 0.6875

Average Precision = (1.0+1.0+1.0+1.0+1.0+1.0+1.0+1.0+0.9+0.9091+0.6875) / 11 = 0.954

The first image in this example is the query image and therefore is not used in calculations. The green ticks are below the images that are similar to the query image, which is a helicopter. This example performed very well as we would continue to find each precision of the correct images until all correct images are found, and therefore will sometimes need to iterate through hundreds of images until all correct images are found to be able to compute the average precision.

Lastly, we also want a metric that gives us a single integer to be able to view the success of the entire system. This metric is called mean average precision and is quite simple to implement as it follows directly from the average precision results. We take the average precisions for each query, in the MPEG7 dataset we used 280 queries, and we find the average of them. The mean average precision (mAP) for the MPEG7 dataset was 0.552, which is not a great result but an expected one as we are still using pixel values which are not a very accurate way for a computer to visualize an image.

Next, I will be looking into feature extraction and how that will improve the results of the system.

**SIFT Features:**

**References:**

Flower image:

* <https://www.google.com/search?q=flower&sxsrf=ALeKk00r8PgsSXvpQ0q6c58qHXAXo5qlag:1625797212059&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjAprTK9tTxAhW07XMBHS35AEIQ_AUoAXoECAIQAw&biw=1044&bih=932#imgrc=rwqXxYAWHAml0M>

Different distance measurements:

* <https://docs.scipy.org/doc/scipy/reference/spatial.distance.html>

Euclidean measurement:

* <https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.euclidean.html#scipy.spatial.distance.euclidean>

Metrics:

* <https://stackoverflow.com/questions/40457331/information-retrieval-evaluation-python-precision-recall-f-score-ap-map?fbclid=IwAR28OIfJGuWAJlPsmwB1sJu9RSGIFtLzKCt-6MHk9E_TEJVS5Yp3AC62vlI>

SIFT:

* <https://stackoverflow.com/questions/51168896/bag-of-visual-words-implementation-in-python-is-giving-terrible-accuracy>
* <https://towardsdatascience.com/bag-of-visual-words-in-a-nutshell-9ceea97ce0fb>
* James Philbin1 , Ondˇrej Chum1 , Michael Isard2 , Josef Sivic1 and Andrew Zisserman. (2007). Object retrieval with large vocabularies and fast spatial matching. <https://www.semanticscholar.org/paper/Object-retrieval-with-large-vocabularies-and-fast-Philbin-Chum/28e4b8ebbdb0e80f03b6f0578deeb38694af081e>
* <https://my.eng.utah.edu/~cs6320/cv_files/ImageMatching.pdf>
* <https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html>
* <https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/>
* <https://datacarpentry.org/image-processing/06-blurring/>

Datasets:

* <https://dabi.temple.edu/external/shape/MPEG7/results.html>
* <https://www.robots.ox.ac.uk/~vgg/data/oxbuildings/index.html>