IMAGE GROUPING

Project Report

EKLAVYA MENTORSHIP PROGRAMME

At

SOCIETY OF ROBOTICS AND AUTOMATION,
VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE
MUMBAI

SEPT-OCT 2021

ACKNOWLEDGMENT

The Image Grouping project was a fantastic opportunity for us both to work closely together on a leading technological field of image processing & clustering. We thoroughly enjoyed the entire process right from the learning stage to the final execution. We gained first hand experience in some of the best python libraries and explored . Our mentors were there to guide & provide their insight to us at each and every step and thanks a ton to SRA for giving us invaluable experience

Special Thanks to our Amazing Mentors:

Mann Doshi
Prathamesh Tagore

Team Members:

Pratham Shah
Email : ppsprathamshah2313@gmail.com

Yash Deshpande
Email : 23yashd@gmail.com

TABLE OF CONTENTS:

1. PROJECT OVERVIEW:

1.1 DESCRIPTION OF USE CASE AND PROJECT:

Segregates images using feature extraction and Clustering algorithms.
The mechanism of this project might get applications in:

- Segregating images with unsupervised learning
- Can be used in your gallery to segregate images.

1.2 ALGORITHM USED:

1.ORB from OpenCV

Oriented FAST and rotated BRIEF (ORB) is a fast robust local feature detector that can be used in computer vision tasks like object recognition or 3D reconstruction. It is based on the FAST keypoint detector and a modified version of the visual descriptor BRIEF (Binary Robust Independent Elementary Features).

2. K-Means from Scikit Learn

K-means is an unsupervised classification algorithm, also called clusterization, that groups objects into *k* groups based on their characteristics. The grouping is done minimizing the sum of the distances between each object and the group or cluster centroid. The distance usually used is the quadratic or euclidean distance.

The algorithm has three steps:

1. Initialization: once the number of groups, *k* has been chosen, *k* centroids are established in the data space, for instance, choosing them randomly.
2. Assignment of objects to the centroids: each object of the data is assigned to its nearest centroid.
3. Centroids update: The position of the centroid of each group is updated taking as the new centroid the average position of the objects belonging to said group.

Link to [github repository]

2. Introduction

2.1 BRIEF IDEA:

This project aims at creating an image grouping algorithm. The algorithm should be able to group similar images on the basis of extracted features. We have used ORB algorithm for extracting features and Scikit K-means clustering algorithm to clusterize images. So it reads images from a folder and applies ORB to all images to give its descriptors and finds optimum no. of groups(K) then applies K-means on descriptors and paste images to their respective cluster folder.

2.2 BASIC PROJECT DOMAINS:

- Image Processing
- Keypoint detection & extraction
- Unsupervised Clustering

3. METHODS AND STAGES OF PROGRESS:

3.1 THEORY AND APPROACH:

Select assorted images of single label test subjects like for example cats & cars. Apply the clustering algorithm to find images of cats in one folder & cars in a seperate folder. It reads images from a folder and applies ORB to all images to give its descriptors and find optimum value of k and applies K-means on descriptors and paste images to their respective cluster folder.

- *Preprocessing*

  To group images, it requires processing the images under test. Image processing is the operation of converting images into computer readable data. To perform the necessary operations we processed the images using the OpenCV python library which allows us to read images in matrix format.

- *Feature extraction via Orb*

  Now we need to extract features from the image to understand the contents of the image. A feature / keypoint is a piece of information about the content of an image; typically about whether a certain region of the image has certain properties.These features are stored in the computer memory in the form of descriptors. The descriptor contains the visual description of the patch and is used to compare the similarity between image features. So, by applying openCV ORB to all images, we stored all keypoints and descriptors of images in the list.

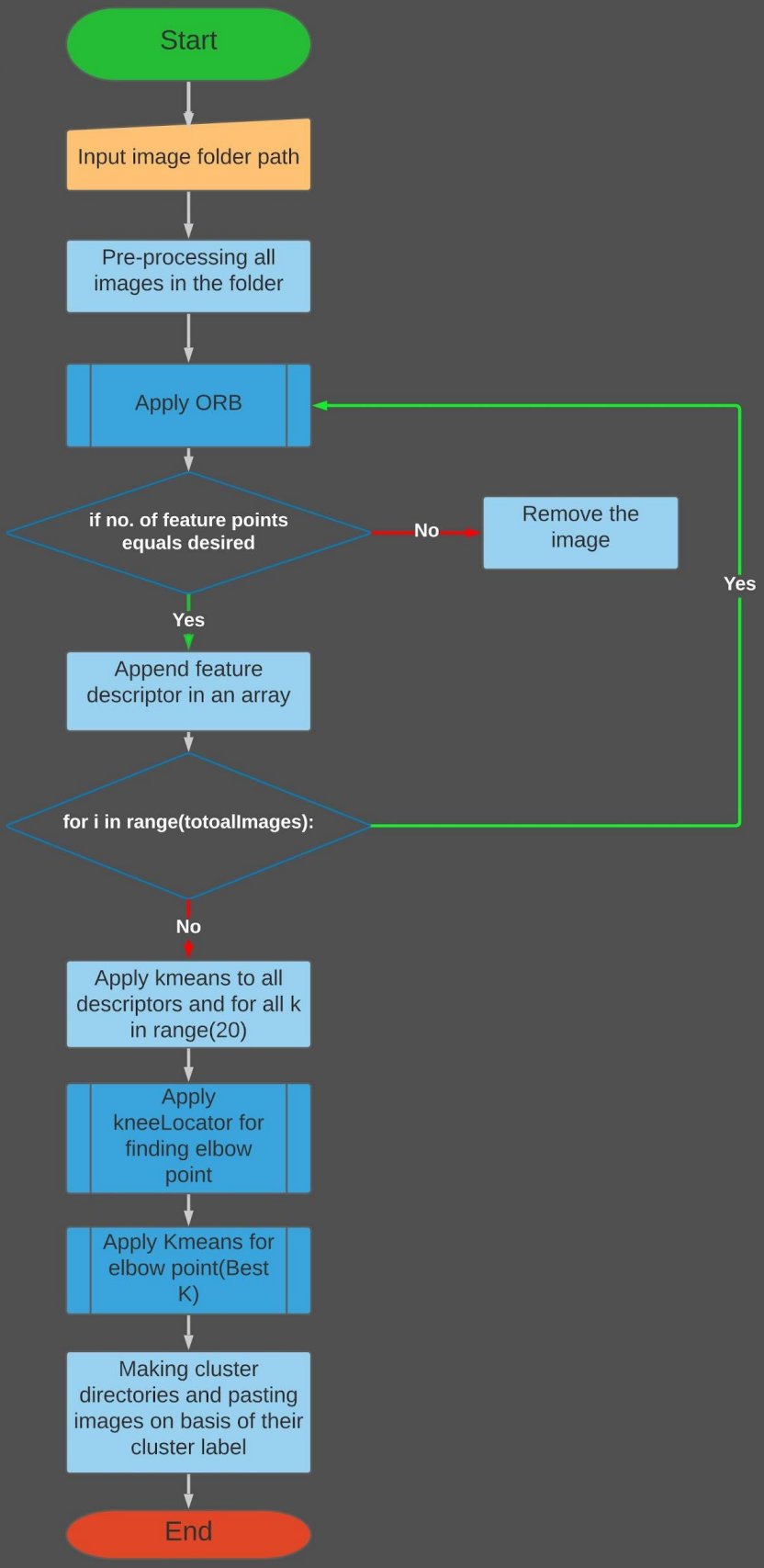
- *K-Means clustering*

  So, after getting descriptors of all images, we need to cluster them by using K-Means clustering. First we need to find no. of clusters so we are doing that by applying the elbow method using distortions. K-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

- *Making directories for cluster and pasting image to its respective directory(cluster)*

  Now we have all images labeled(cluster number), we can group them by making separate directories using os library and copy paste images from the main folder to their respective directories by shutil.

Code WorkfLow



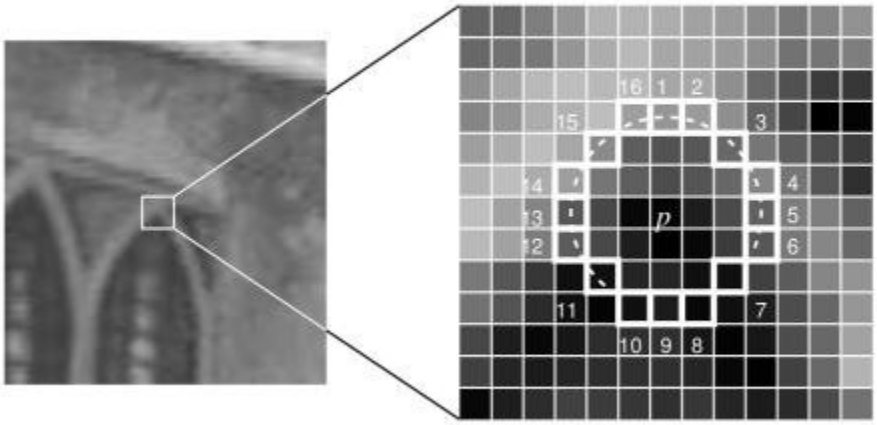
3.2 Feature extraction

ORB performs as well as SIFT on the task of feature detection (and is better than SURF) while being almost two orders of magnitude faster. ORB builds on the well-known FAST keypoint detector and the BRIEF descriptor. Both of these techniques are attractive because of their good performance and low cost. ORB's main contributions are as follows:

- The addition of a fast and accurate orientation component to FAST
- The efficient computation of oriented BRIEF features
- Analysis of variance and correlation of oriented BRIEF features
- A learning method for decorrelating BRIEF features under rotational invariance, leading to better performance in nearest-neighbor applications

Fast(Features from Accelerated and Segments Test)

Given a pixel p in an array fast compares the brightness of p to surrounding 16 pixels that are in a small circle around p. Pixels in the circle are then sorted into three classes (lighter than p, darker than p or similar to p). If more than 8 pixels are darker or brighter than p then it is selected as a keypoint. So keypoints found by fast gives us information of the location of determining edges in an image.



However, FAST features do not have an orientation component and multiscale features. So the orb algorithm uses a multiscale image pyramid. An image pyramid is a multiscale representation of a single image, consisting of sequences of images all of which are versions of the image at different resolutions.

Brief(Binary robust independent elementary feature)

Brief takes all keypoints found by the fast algorithm and converts it into a binary feature vector so that together they can represent an object. Binary features vector also know as binary feature descriptor is a feature vector that only contains 1 and 0. In brief, each keypoint is described by a feature vector which is 128–512 bits string.

Brief start by smoothing the image using a Gaussian kernel in order to prevent the descriptor from being sensitive to high-frequency noise. Then brief select a random pair of pixels in a defined neighborhood around that keypoint. The defined neighborhood around a pixel is known as a patch, which is a square of some pixel width and height. The first pixel in the random pair is drawn from a Gaussian distribution centered around the keypoint with a stranded deviation or spread of sigma. The second pixel in the random pair is drawn from a Gaussian distribution centered around the first pixel with a standard deviation or spread of sigma by two. Now if the first pixel is brighter than the second, it assigns the value of 1 to corresponding bit else 0.

ORB specifies the rBRIEF algorithm as follows:

1) Run each test against all training patches.

2) Order the tests by their distance from a mean of 0.5, forming the vector T.

3) Greedy search:

- Put the first test into the result vector R and remove it from T.
- Take the next test from T, and compare it against all tests in R. If its absolute correlation is greater than a threshold, discard it; else add it to R.
- Repeat the previous step until there are 256 tests in R. If there are fewer than 256, raise the threshold and try again

rBRIEF shows significant improvement in the variance and correlation over steered BRIEF.

Code implementation

```
orb = cv2.ORB_create(nfeatures= totalFeaturePoints, edgeThreshold=0,fastThreshold=0)
  temp_kp, temp_des = orb.detectAndCompute(allImages[i], None) #Gets keypoints and descriptors
```

3.3 K-means

The K-means algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the *inertia* or within-cluster sum-of-squares. It is a numerical, unsupervised, non-deterministic, iterative method. It has to calculate the distance between each data object and all cluster centers in each iteration, which makes the efficiency of clustering is not high.

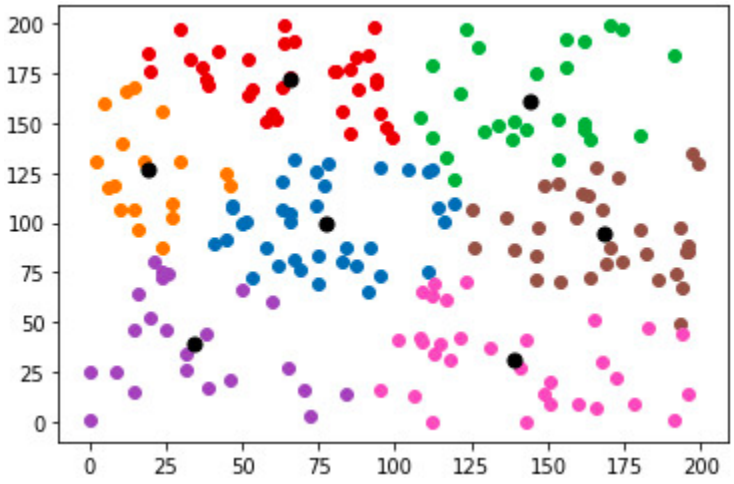
The algorithm consists of two separate phases. The first phase selects k centers randomly, where the value k is fixed in advance. The next phase is to take each data object to the nearest center. Euclidean distance is generally considered to determine the distance between each data object and the cluster centers. When all the data objects are included in some clusters, the first step is completed and an early grouping is done. Recalculating the average of the early formed clusters. This iterative process continues repeatedly until the criterion function becomes the minimum.

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} \left| x - x_i \right|^2$$

E is the sum of the squared error of all objects in the database. The distance of criterion function is Euclidean distance, which is used for determining the nearest distance between each data object and cluster center. So, the process of the k-means algorithm is :- Input: Number of desired clusters, k, and a database D={d1, d2,…dn} containing n data objects. Output: A set of k clusters Therefore the computational time complexity of the k-means algorithm is O(nkt),where n is the number of all data objects, k is the number of clusters, t is the iterations of the algorithm and the positive integer t is known as the number of k-means iterations. The precise value of t varies depending on the initial starting cluster centers, usually requiring k << n and t << n.

Results of our kmeans code(From Scratch)

(Black point is cluster centroid)



Code implementation

```
distortions = []
inertias = []
mapping1 = {}
mapping2 = {}
K = range(1, 20) #Kept a limit of 20 clusters, so it will predict no. cluster from 1 to 20

'''
  Distortion: It is calculated as the average of the squared distances from the cluster centers of the respective clusters. Typically, the Euclidean distance metric is used.
  Inertia: It is the sum of squared distances of samples to their closest cluster center.

We iterate the values of k from 1 to 20 and calculate the values of distortions for each value of k and calculate the distortion and inertia for each value of k in the given range.
'''

for k in K:
    # Building and fitting the model
    kmeanModel = KMeans(n_clusters=k, random_state=0, ).fit(X)
    # kmeans.labels_
    distortions.append(sum(np.min(cdist(X, kmeanModel.cluster_centers_,
                                        'euclidean'), axis=1)) / X.shape[0])
    inertias.append(kmeanModel.inertia_)

    mapping1[k] = sum(np.min(cdist(X, kmeanModel.cluster_centers_,
                                   'euclidean'), axis=1)) / X.shape[0]
    mapping2[k] = kmeanModel.inertia_
k = []

for key, val in mapping1.items():
    k.append(val) #stroing all values of distortions

x = range(1, len(k)+1)
kn = KneeLocator(x, k, curve='convex', direction='decreasing') #finding elbow point using Distortion
k = kn.knee
if k == None:
  k = int(input("You need to enter no. of cluster, our code couldnt predict :( ")) #It couldn't predict cluster when datasets contains similar images
print("No. of clusters: ", str(k))
kmeans = KMeans(n_clusters=k, random_state=0, ).fit(X) #SkLearn Kmeans
```

4. PROJECT CONCLUSION AND FUTURE WORK:

4.1 EFFICIENCY AND CURRENT ACCURACY:

Task	Time Taken (in seconds)
Pre-processing	5.941
Feature Extraction	48.796
Finding no. of clusters (Optimum k)	19.147
Clustering	0.874
Creating Directories and pasting images	0.757

Did this for 101 images and tested on Lenovo Legion 5(Ryzen 5 4600H) and got accuracy close to 80%. All of our test cases and related code can be found in the project github repository.
We tried on our K-Means code(from scratch) and got accuracy around 55-65%.

4.2 What did we achieve :

- We learnt basics of openCV and implemented it for reading images and used its ORB for feature extraction
- Used Scikit learn Python library to perform unsupervised clustering algorithm on the obtained dataset descriptors
- Learnt about file management & directories for Clusters(Groups) in python using OS library
- Integrated all above 3 code, so it reads images from a folder and applies ORB to all images to give its descriptors and applies K-means on descriptors and paste images to their respective cluster folder.

4.3 FUTURE ASPECTS OF PROJECT:

- Improving accuracy
- Finding optimum of K
- Make a web app where one can upload images and can cluster it online.
- We are currently learning Machine learning and neural networks which will help to label our cluster at least for some basic and common objects).

5. References:

KMeans Resources:
Computerphile youtube video
https://realpython.com/k-means-clustering-python/
https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a
https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/
https://medium.com/data-folks-indonesia/step-by-step-to-understanding-k-means-clustering-and-implementation-with-sklearn-b55803f519d6
https://www.kdnuggets.com/2019/05/guide-k-means-clustering-algorithm.html
https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

ORB Resources:
https://medium.com/data-breach/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf
https://www.geeksforgeeks.org/feature-matching-using-orb-algorithm-in-python-opencv/