Author: Kunjan Mhaske

# ------------------------------------BAG OF WORDS MODEL----------------------------------------

**SIFT: Scale-Invariant Feature Transform**

To overcome the scaling and orientation change problem while detection of corners and edges, Scale-Invariant Feature Transform is introduced which extracts the keypoints and descriptors. There are mainly 4 steps involved in SIFT algorithm:

- Scale space extrema detection: SIFT uses Difference of Gaussians that is obtained as difference of gaussian blurring of an image with two different sigmas.
- Keypoint Localization: Once potential keypoints are found, sift eliminates any low-contrast keypoints and edge keypoints. Remaining are the strong interest points.
- Orientation Assignment: Orientation is assigned to achieve the invariance to image rotation. Gradient magnitude and direction is calculated of the surrounding region according to the scale.
- Keypoint Descriptor: 16*16 neighbourhood around the keypoint is taken which contains 16 sub blocks of 4*4 sizes. For each sub-block 8 bin orientation histogram is created. Hence 128 bin values are available in keypoint descriptor.

**Keypoint**: It is the circular image region with its respective orientation.

It is geometric frame of four parameters:

- The keypoint center
- Coordinates x and y
- Its scale (radius of region)
- Orientation (angle expressed in radians)

**Descriptor:** It is a 3-D spatial histogram of the image gradients in characterizing the appearance of a keypoint. The gradient of each pixel is regarded as a sample of 3-D elementary feature vector formed by pixel location and gradient orientation. Orientation are quantized into eight bins and the spatial coordinates into 4 each.

The dimension of descriptor is as follows: Number_of_Keypoints * 128

**Extract SIFT Features:**

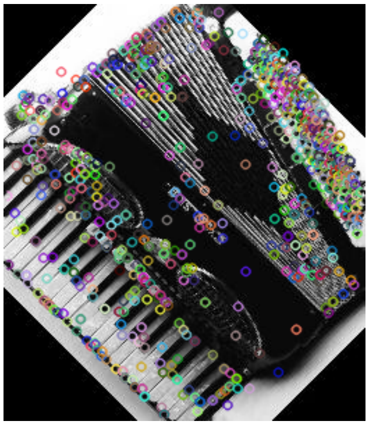
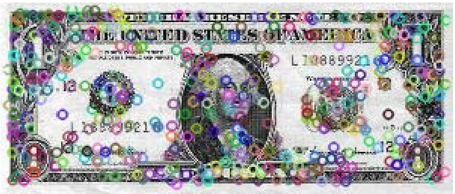
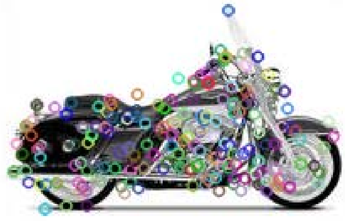
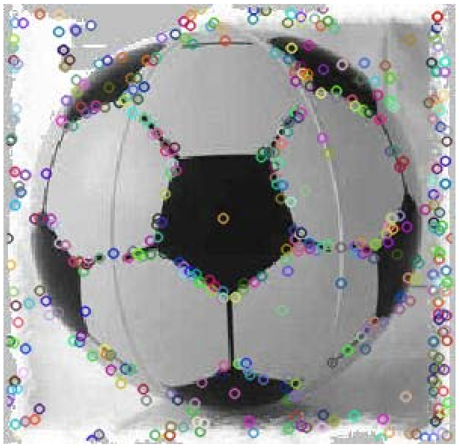
Step 1: Read the images in grayscale format.

Step 2: Use SIFT from opencv-contrib-python = 3.4.2.17

Step 3: Create object of SIFT: sift = cv2.xfeatures2d.SIFT_create()

Step 4: Extract the keypoints and descriptors from the given image:
keypoints, descriptors = sift.detectAndCompute(image)

Step 5: Plot the keypoints on the given image at their respective position using following function: cv2.drawKeypoints(image,keypoint)

| Category | Input Image | Output Image |
|---|---|---|
| Accordion | |  |
| Dollar Bill | |  |
| Motorbike | |  |
| Soccer Ball | |  |

**Matching Keypoints:**

Step 1: Take BruteForceMatcher object

```
bfm = cv2.BFMatcher_create(cv2.NORM_L2, crossCheck=True)

# NORM_L1 = Manhattan distance

# NORM_L2 = Euclidean distance
```

Step 2: Match the descriptors

```
match = bfm.match(img1desc, img2desc)
```

Step 3: Sort the matches according to distances

```
match = sorted(match, key=lambda i: i.distance)
```

Step 4: Draw top 20 matches of keypoints with one image to another image of dataset

```
match_img = cv2.drawMatches(img1, img1keyp, img2, img2keyp, match[:20], img2.copy(), flags=0)
```

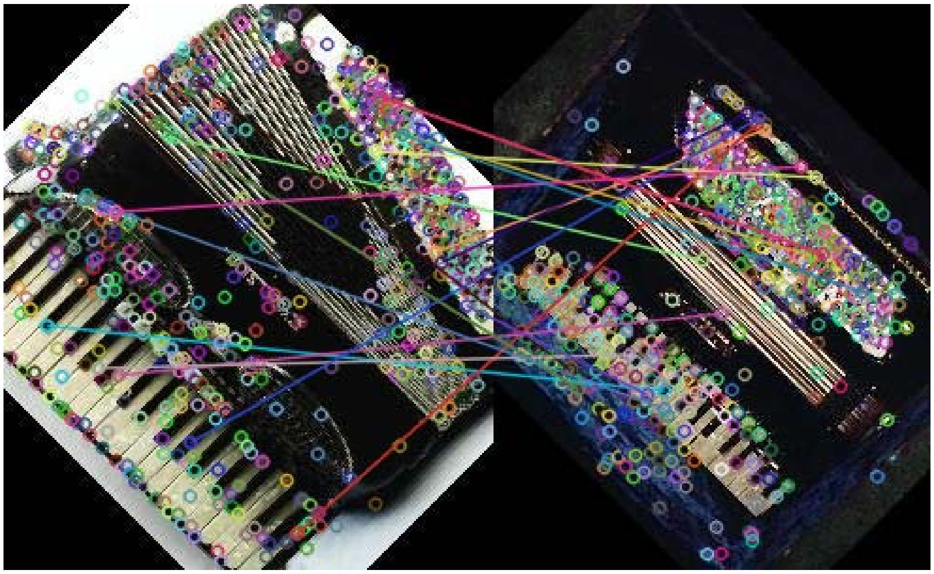


Fig. Accordion keypoint matches between image 1 and image 2

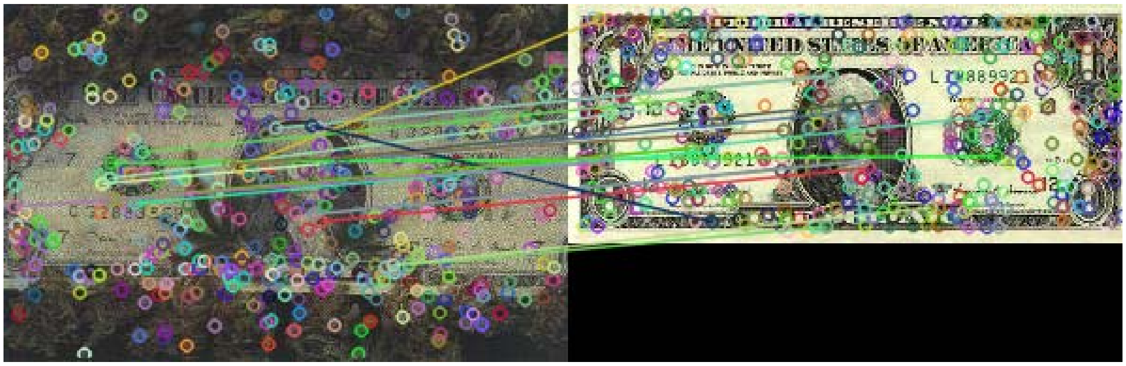


Fig. Dollar bill keypoint matches between image 1 and image 2

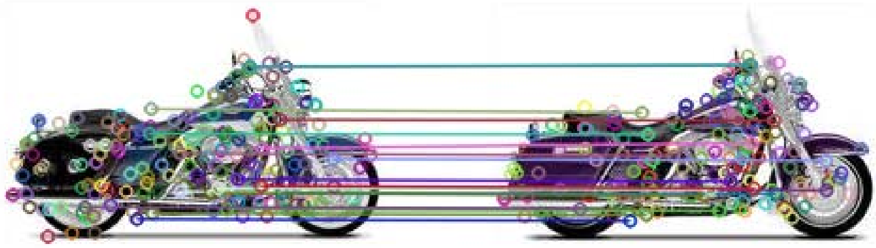


Fig. Motor bike keypoint matches between image 1 and image 2

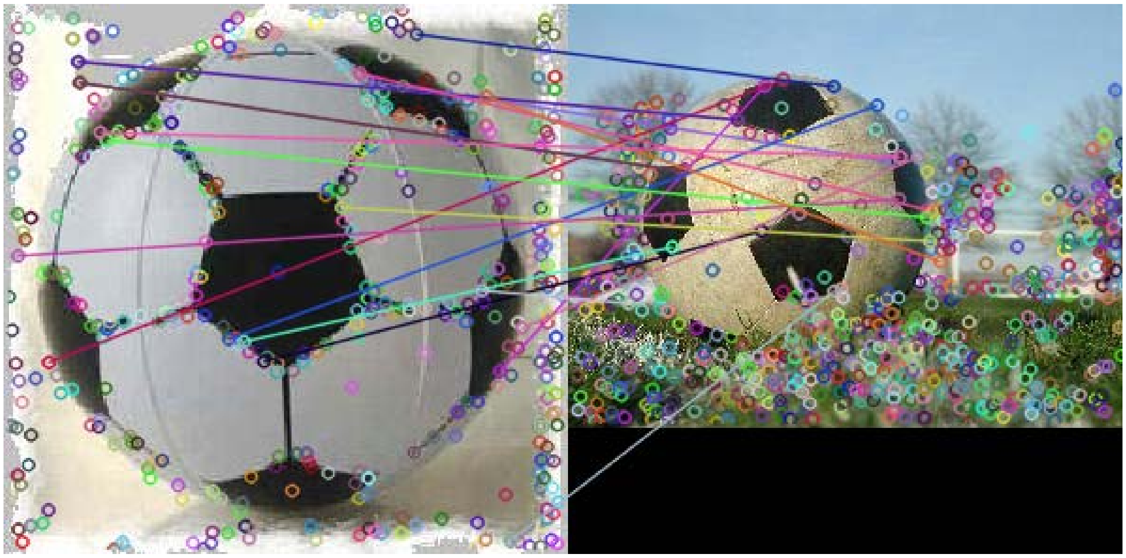


Fig. Soccer ball keypoint matches between image 1 and image 2

**KMeans Clustering:**

It is a method of creating the K number of clusters by grouping the points based on the neighborhood of each K points. When multiple points are near by each other and respective k point, then we could call them as one cluster. It gives you the label for each cluster point and the center of the respective cluster. The K points can be assigned randomly at initial stage and then rearrange it according to the calculation of the nearby points distance.

**Bag of Words Model:**

In Bag of Words model, the image features are treated as words and the bag of words is a vector of occurrence counts of a vocabulary of local image features.

Features: Scale Invariant Feature Transform (SIFT) – 128 dimension vector of descriptor

Clustering method: KMeans clustering from sklearn

**Create of Bag of Words:**

Step 1: Create the training labels list according to the category of the given images to map the output of clusters into their respective category.

Step 2: Create Vertical Stack of all descriptors of images from SIFT output to feed into KMeans function.

Step 3: Create Object of KMeans with given number of clusters. Lets take number of clusters be 100 and create object as follows: KMeans_obj = KMeans(n_clusters=100)

Step 4: Call fit_predict() method to get the keypoints descriptors after assigning them to respective cluster centers. fit_predict(KMeans_obj,vStack) computes the cluster center and predict the cluster index for each sample in the given stack of descriptor.

Step 5: Create the vocabulary of the words which is the set of given features which describes an image individually. It is described as n_clusters * n_images. Hence, locate the cluster that contains the respective feature i.e. Cluster number whose cluster centroid is closer to the location of current feature and assign that cluster number to respected feature.

**Output:-**

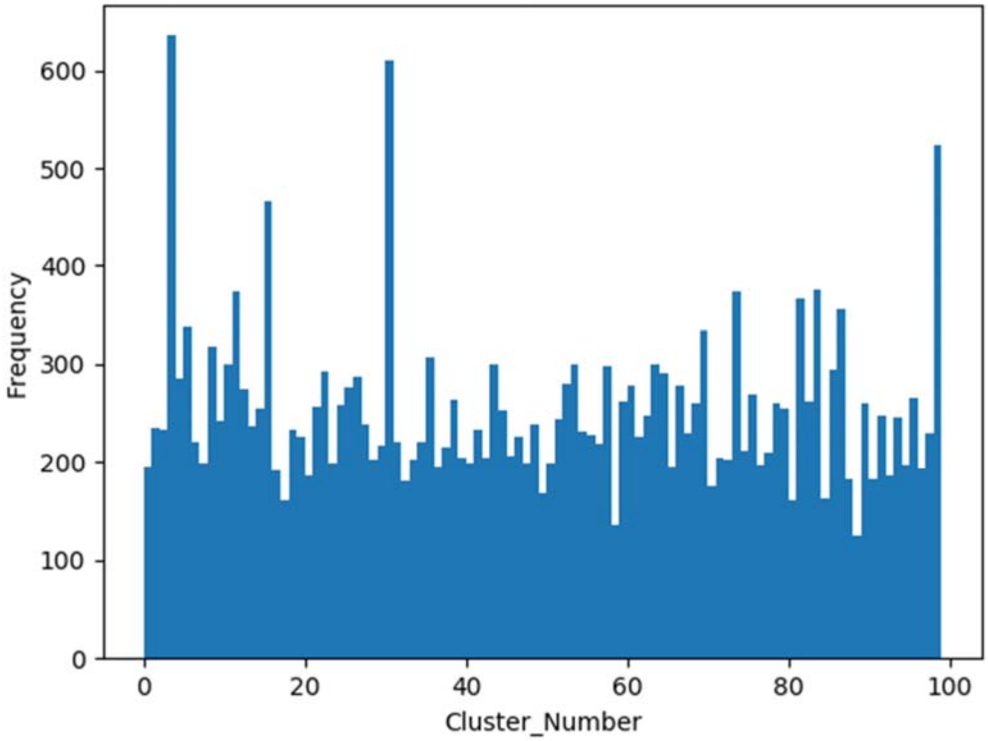
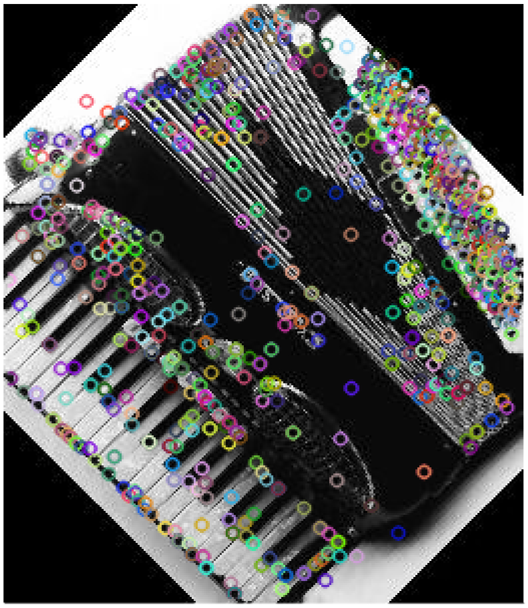


Fig: Histogram of cluster points of bag of words of all images from 4 categories (#clusters = 100)



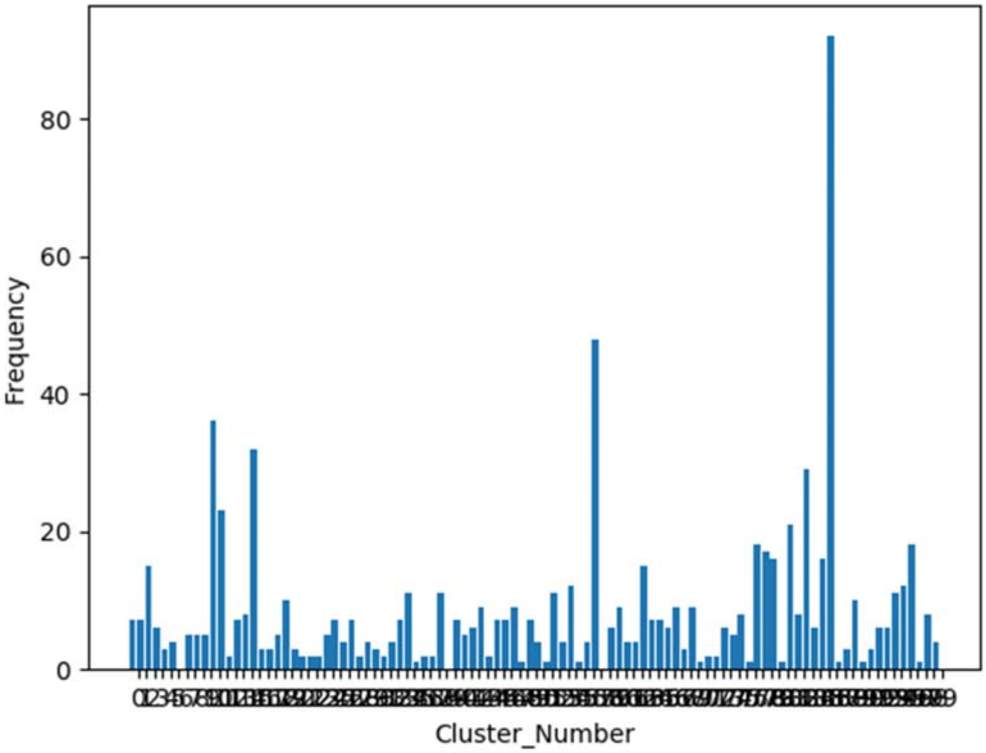


Fig. Histogram for Accordion

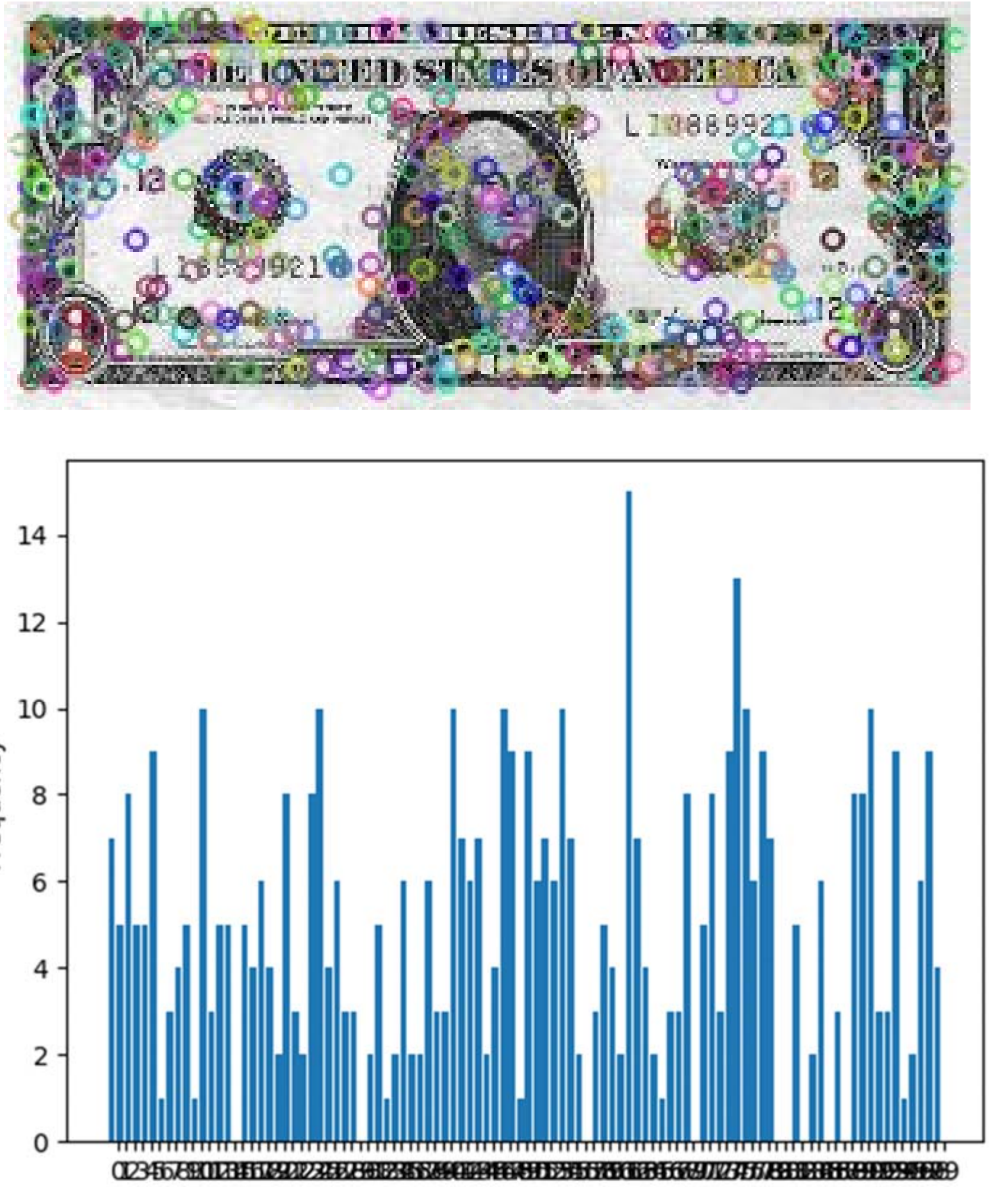
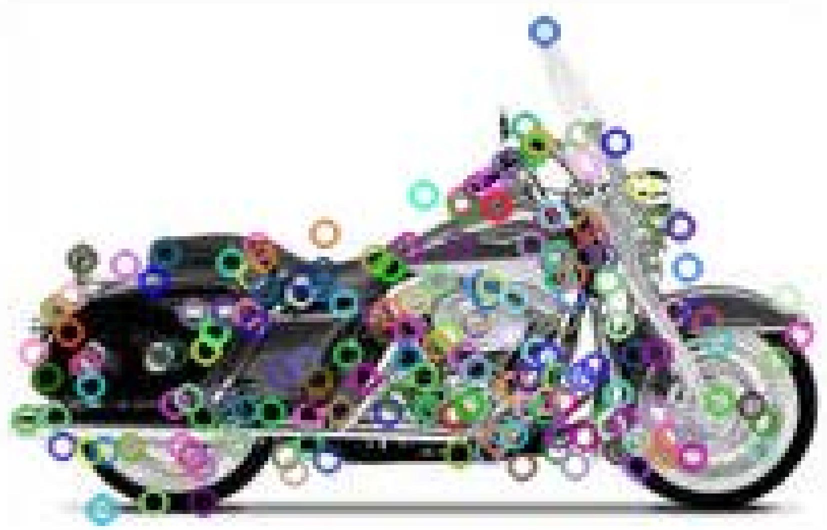


**Fig. Histogram for Dollar Bill**



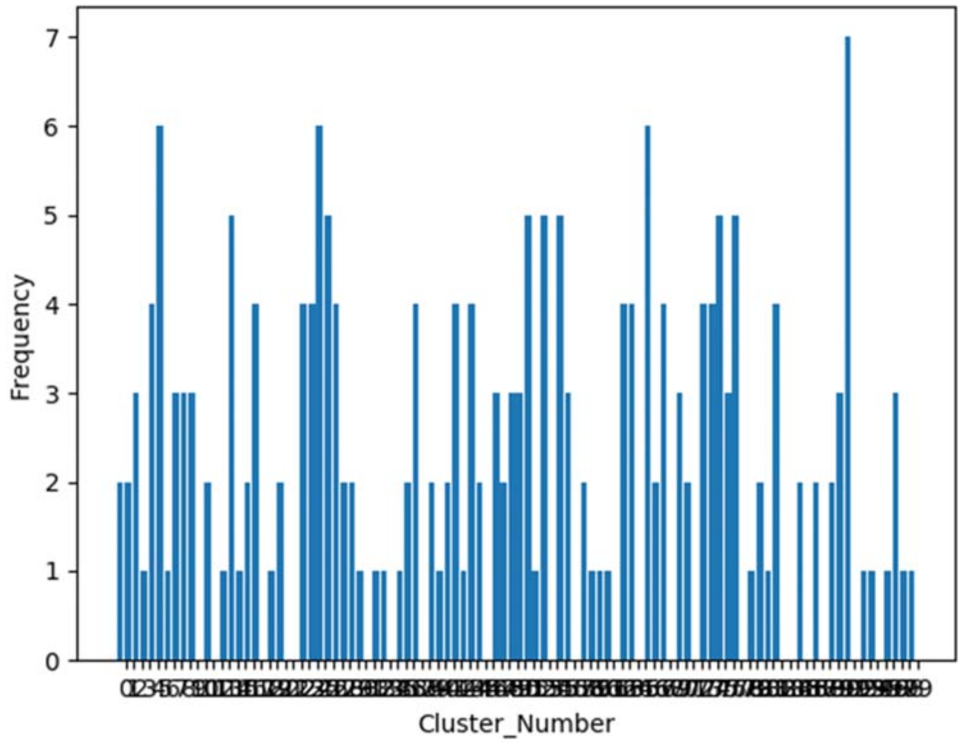
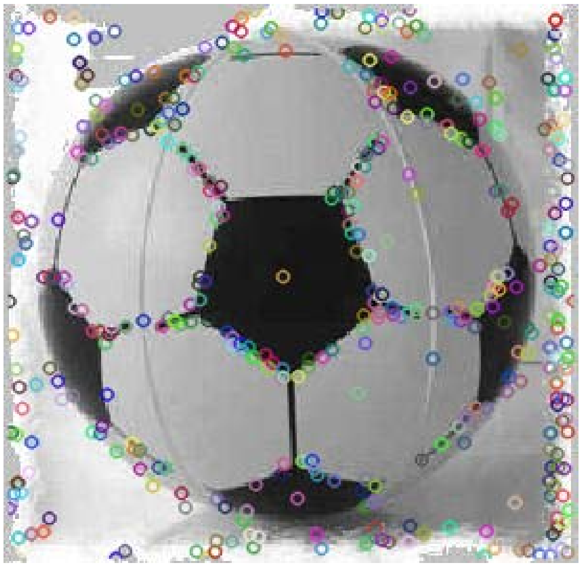


**Fig. Histogram for Motor bike**



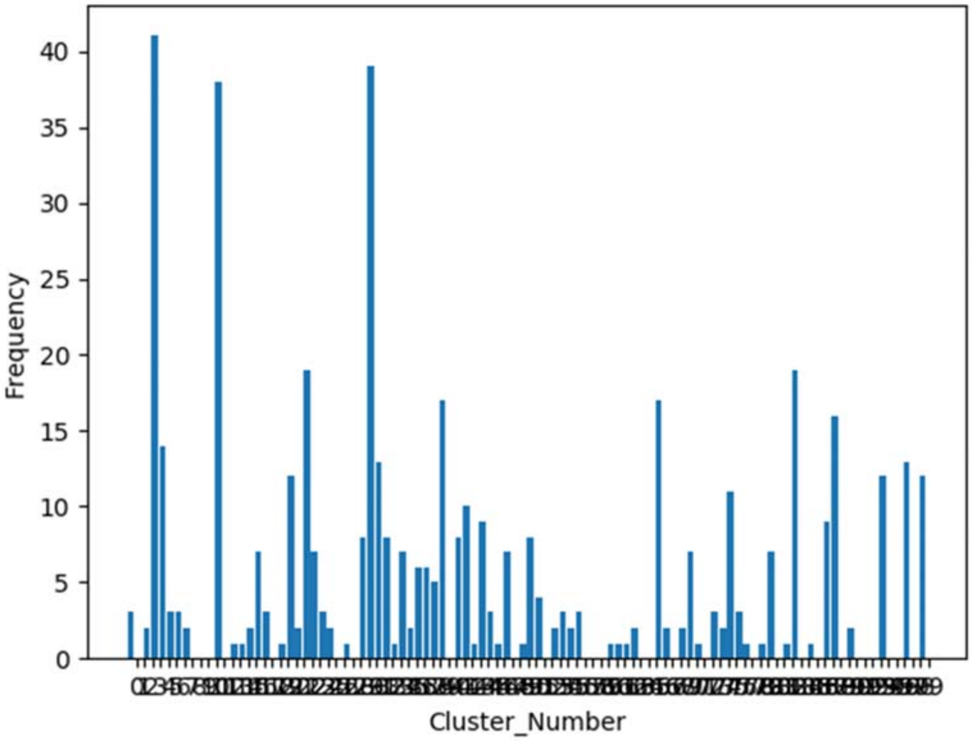


**Fig. Histogram for Soccer Ball**