Lab 1: Bag-of-Features representation for Content Based Image Retrieval

Thomas Wimmer

October 2022

1 Detectors and descriptors

1.1 ORB

The ORB method [3] uses an extension of the FAST detector that calculates the FAST detector at several different resolutions (scales) of the image and thus assigns a characteristic scale to all detected keypoints. The FAST detector marks points as keypoints if their circular neighborhood has long contiguous patterns with significantly brighter values than the point itself. The characteristic orientation for a keypoint is the direction between the centre point and the centre of mass (brightness values) of its local neighborhood.

The ORB method uses a rotated version of the BRIEF descriptor method that assigns descriptive vectors to the detected keypoints. The descriptive variables take integer values between 0 and 255 and the dimension of the descriptors is 32.

1.2 Kaze

The Kaze method [1] detects keypoints in a nonlinear scale space (i.e. anisotropic diffusion instead of Gaussian scale space) and is based on efficient additive operator splitting. The detected keypoints are the points with the maximum scale-normalised determinants of the local Hessian matrix. For each of these keypoints, a dominant orientation is calculated in a similar way to SURF.

The description of the detected points is done with a modified M-SURF descriptor. First order derivatives in the local neighborhood of the keypoints are computed, weighted and summed to a descriptor vector and rotated according to the dominant orientation. The final descriptor vector has a length of 64 with values in the range $[0, 1]$.

A visual comparison between the detected keypoints with the two methods can be found in Figure 1.

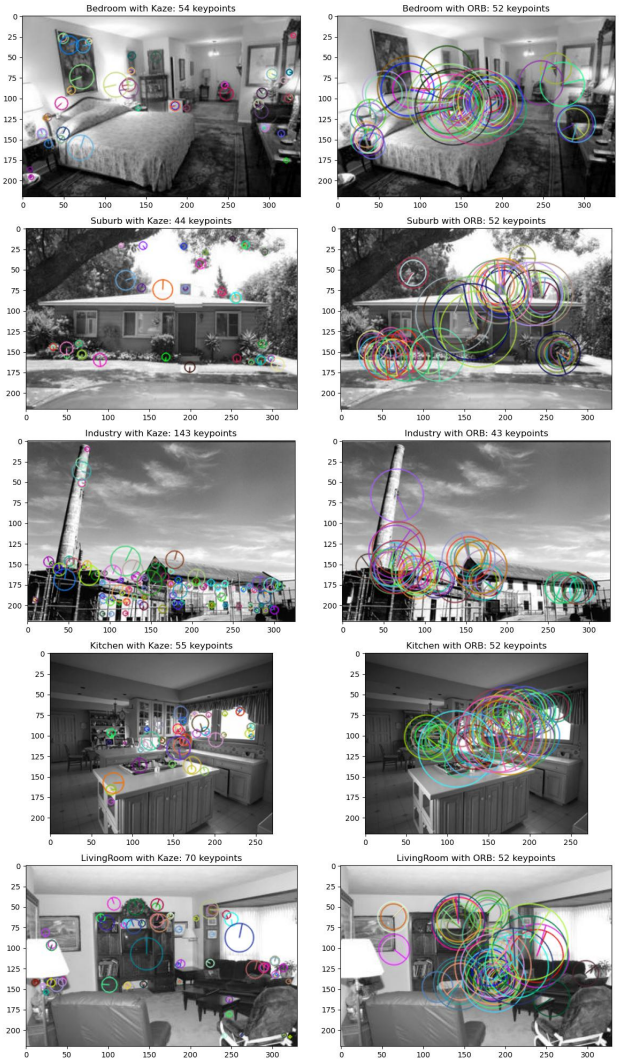


Figure 1: Comparison between the detected keypoints with Orb and Kaze. The different distribution of the detected keypoints with the two methods is striking. While Kaze identifies different keypoints that are well distributed across the image, ORB sometimes detects the same or very close keypoints several times. In particular, the keypoints detected by ORB are almost all clearly identifiable as corner points. Using only this visual analysis, it can be surmised that the keypoints detected with Kaze are better suited to identifying a scene (our task in this assignment) than the corner points identified by Orb. These would presumably be more suitable for determining correspondences and estimating structure and motion.

2 Codebook construction

The rule of thumb that is usually used to determine the size of the codebook (i.e. the number of clusters in the K-Means method) is the so-called "elbow method". This heuristic method approximates the K at which the loss of the K-Means method stagnates and decreases only slightly for $K' > K$ (usually a small linear decrease), while for $K' < K$ the loss still decays significantly as K' is increased. This technique should theoretically find the best number of clusters that describes the data set well, but is not overfitted to the data.

In practice, however, it often happens that no sharp "elbow" can be identified and other heuristics (such as a threshold for the ratio between intra- and inter-cluster variance) have to be used.

The K-Means implementation of `sklearn` was used to compute the clusters and its centres. This implementation uses the euclidean distance as distance metric between the points which seems reasonable as this guarantees the convergence of the K-Means algorithm in comparison to e.g. the Mahalanobis distance [2].

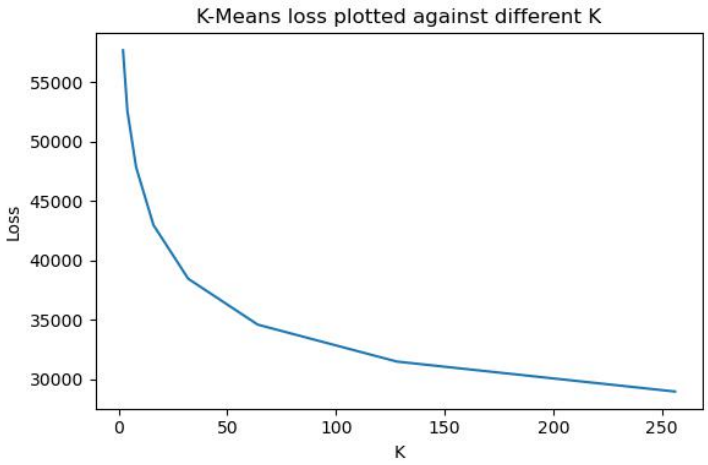


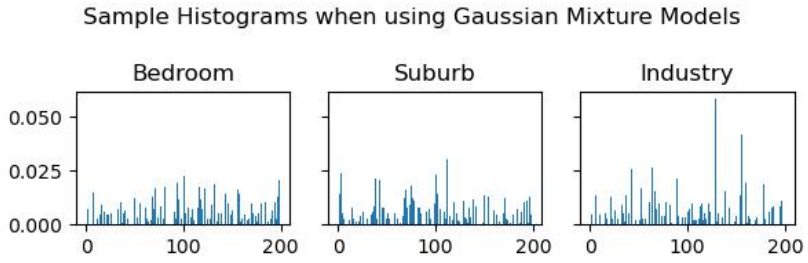
Figure 2: Plotting the final loss of the K-Means cluster against various (exponentially) sampled values for K, one can use the elbow method to find that the loss for $K \approx .40$ more or less terminates its exponential decay.

Using the "elbow-method" method, we can identify an optimal value for the number of words in the codebook around 80 to 100.

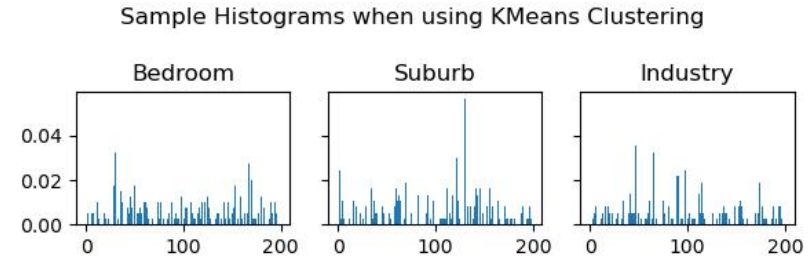
However, there is another method we can use to determine the number of words we want in our codebook by estimating the number of different key point types in the different scenes. Using this rule of thumb, we can estimate that there are about 200 to 300 keypoints (≈13-20 per class) that might be interesting to distinguish in the codebook and thus in the global descriptors. Since computational resources are sparse and clustering takes a lot of time, the

codebook was set to a size of 200 in the following experiments.

3 Indexing the Test dataset



(a) Example histograms when using Gaussian mixture models for the codebook and summing the cluster probabilities over all detected key points as aggregation operator



(b) Example histograms when using K-Means clustering for the codebook and calculating the histogram by simply counting the nearest clusters for all detected keypoints.

Figure 3: Sample Histograms for different indexing methods.

Indexing of the test dataset can be done by applying the same key point and descriptor detection method as before for the training samples and matching the found points with the previously determined codebook (i.e. searching for the closest cluster for all detected key points). With the clusters found, it is easy to create a histogram by counting the number of occurrences of each cluster in the image. This method corresponds to the idea outlined in the slides of the lecture on "Multiscale Feature Extraction and Description" (p.54).

Of course, more sophisticated aggregation methods could also be used, e.g. involving the "certainty" of the association of points to the clusters. This method, however, requires a different codebook construction with e.g. a Gaussian mixture model. In order to improve the results, this method was implemented to be able to compare the two different methods.

Using a Gaussian mixture model to build our codebook, which is essentially very similar to soft K-Means clustering, we can not only use the nearest cluster

for each new keypoint, but instead use the probability of the point belonging to a cluster for all clusters. In this way, we can simply sum the probability distributions for all detected keypoints in a new image and obtain a histogram for the unseen image. Examples of the global descriptors / histograms for some images from different classes are shown in Figure 3.

In order to get rid of possible side effects through different amounts of detected keypoints in different images, it is possible to normalize the histograms for all images. In the experiments that were carried out for the lab, both cases were considered and compared.

4 Image Retrieval

Having the codebook and the method of indexing for the images defined, it is easy to create an image retrieval method. Using the indexing method, all test images are mapped to histograms (the global descriptors).

We can now find the nearest neighbors of an image in the test set by using a K-Nearest-Neighbor algorithm. However, since the implementation of the kd-tree only allows Minkowski distances, a brute-force k nearest neighbor search was implemented. The nearest neighbors should be the most similar pictures to the input image. Of course, the nearest neighbor is the image itself, so when referring to the top-1 and the top-3 nearest neighbors in the following section, the nearest neighbor(s) besides the original image are meant.

4.1 Difference Metrics for Image Retrieval

Since the histograms / global descriptors are essentially just vectors, we can apply several different metrics to compute the distance between two global descriptors. The distance metrics used in the experiments are:

- χ^2 distance: $\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{(x_i + y_i)}$
- Histogram intersection: $\sum_{i=1}^{d} \min(x_i, y_i)$
- Bray-Curtis distance: $\frac{\sum_{i=1}^{d} |x_i - y_i|}{\sum_{i=1}^{d} |x_i + y_i|}$
- Canberra distance: $\sum_{i=1}^{d} \frac{|x_i - y_i|}{|x_i| + |y_i|}$
- Correlation distance: $1 - $ `np.corrcoef(x, y)[0, 1]`
- Euclidean distance (L_2): $\sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$
- Hellinger distance: $\sqrt{\frac{1}{2} \sum_{i=1}^{d} (\sqrt{x_i} - \sqrt{y_i})^2}$

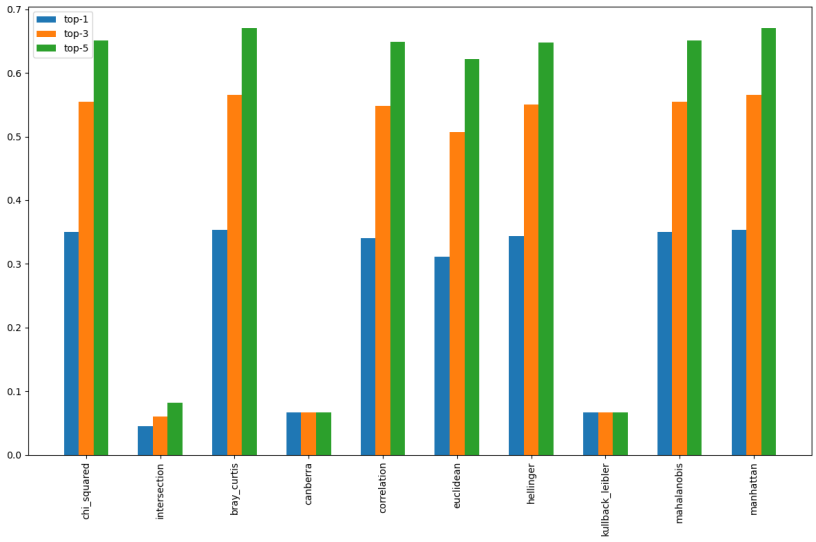
- Kullback-Leibler distance: $\sum_{i=1}^{d} x_i \log \frac{x_i}{x_2}$
- Mahalanobis distance: $\sqrt{\sum_{i=1}^{d} \frac{(x_i - y_i)^2}{x_i + y_i}}$
- Manhattan distance (L_1): $\sum_{i=1}^{d} |x_i - y_i|$

5 Evaluation and Results

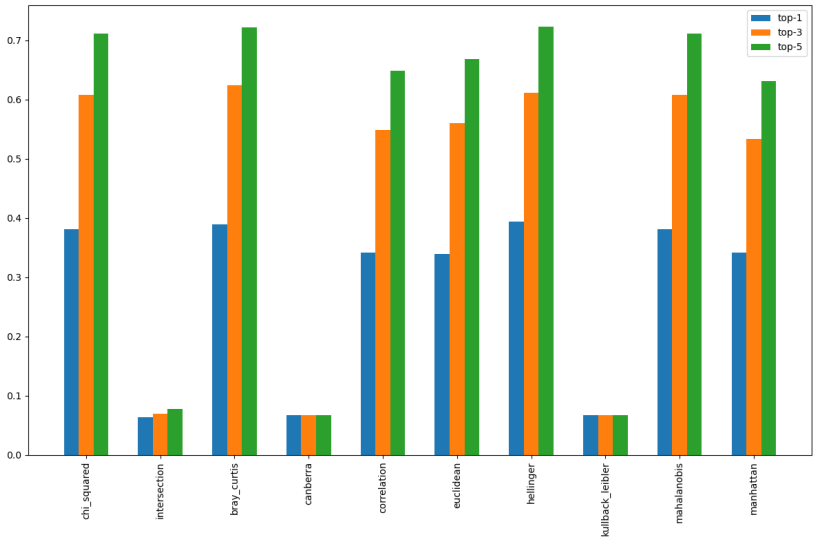
For the evaluation of the results, we can conveniently use the given labels of the images to check whether the labels of the nearest neighbor(s) match the label of the test sample. For the evaluation, the number of top-1, top-3 and top-5 matches was evaluated, i.e. the average match with the closest neighbor (different from query image) in the test set and whether the class of the query image is included in the classes of the nearest 3/5 neighbors.

Through an extensive hyperparameter search, the hyperparameters were set to use the `Kaze` method with the following parameters: `upright=False, threshold=0.0003, nOctaves=5, nOctaveLayers=3, diffusivity=3`. Using the computed descriptors, one can construct the codebook and index the test dataset to find the results for the different methods and distance measures in the K-Nearest Neighbor algorithm. The results for the different metrics are displayed in Figures 4, 5.

As can be seen from the diagrams, the best metric differs for the different methods used to create the histogram. For non-normalized histograms, the best results are obtained with the Hellinger, Mahalanobis, χ^2 or the Bray-Curtis distance. For normalized histograms, the Manhattan distance usually outperforms the other distance metrics, but the Bray-Curtis and Mahalanobis distances also perform well. The best overall results were obtained by combining the non-normalized histogram from the Gaussian Mixture Model Codebook and the Hellinger distance (top-1: **0.45**, top-3: 0.67, top-5: 0.76). It can be observed that in both cases (both when creating the codebook from K-Means and from the Gaussian Mixture Model), the best results when using the non-normalized histograms outperform the best results when using the normalized histograms. It is also observed that using the Gaussian Mixture Model to create the codebook significantly improves the results.

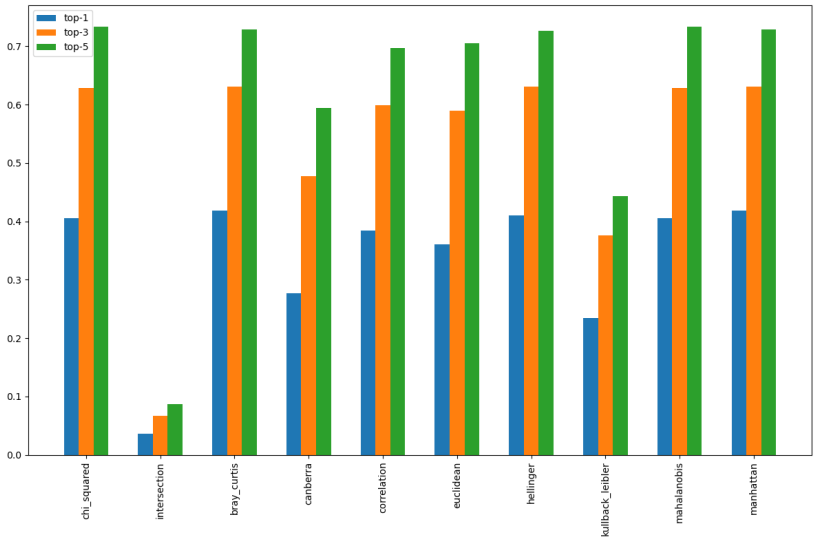


(a) Codebook: KMeans Clustering, Normalized Histograms

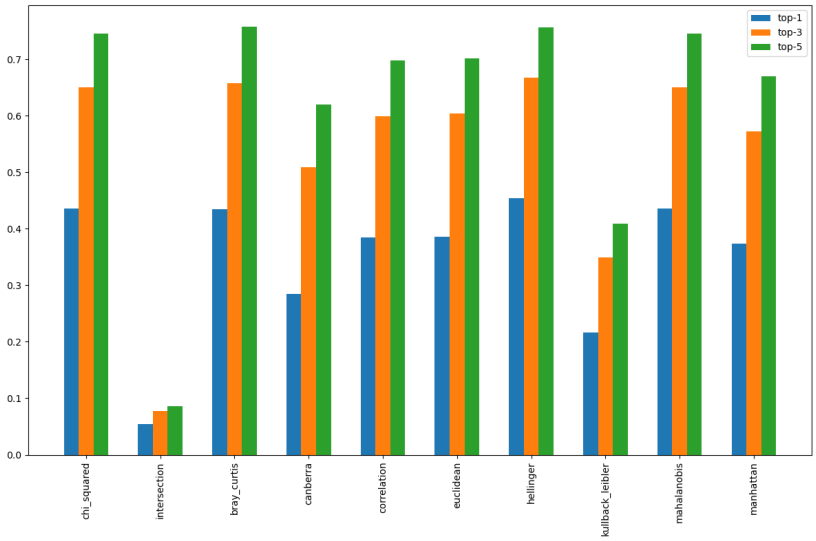


(b) Codebook: KMeans Clustering, Non-normalized Histograms

Figure 4: Agreement of the classes of retrieved images compared to the query image for the creation of the codebook using the K-Means algorithm and using different distance metrics.



(a) Codebook: Gaussian Mixture Model, Normalized Histograms



(b) Codebook: Gaussian Mixture Model, Non-Normalized Histograms

Figure 5: Agreement of the classes of retrieved images compared to the query image for the creation of the codebook using Gaussian Mixture Models and using different distance metrics.

It is also interesting to check the confusion matrix for the best model that could be found. As can be seen in Figure 6, the method is very good in finding matching images for suburbs (1), coastal scenes (5), forests (6) and highways (7) but often confuses kitchen (3) and living room (4) scenes which can easily explained by the similarity of these scenes. One can clearly detect that the scenes inside buildings (bedroom (0), kitchen (3), living room (4), and office (13)) often have similar keypoints and are thus more confused than other classes. However, the overall accuracy of this retrieval method is surprisingly high.

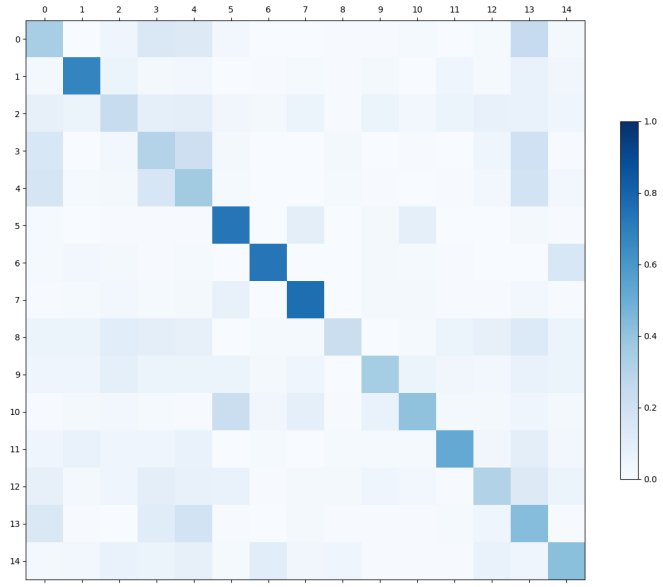


Figure 6: Confusion matrix for the top-1 nearest neighbor for the different classes in the test set. Rows correspond to the real labels and columns to the predicted labels.

Finally, in Figure 7, a sample output of the constructed system can be found for one example of all the different classes. The left column in the image shows the test image and its given class, while the three columns to right show the three nearest neighbors found in the test dataset.

Figure 7: Sample outputs of the constructed system. Left column corresponds to query image and the three columns to right show the three nearest neighbors in the test set.

References

[1] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J Davison. Kaze features. In *European conference on computer vision*, pages 214–227. Springer, 2012.

[2] Itshak Lapidot. Convergence problems of mahalanobis distance-based k-means clustering. In *2018 IEEE International Conference on the Science of Electrical Engineering in Israel (ICSEE)*, pages 1–5. IEEE, 2018.

[3] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.