Report for Week 3

k-means & Bag-of-Visual words
or how to create a Google Goggles-like system

Patrick de Kok (5640318) Georgios Methenitis (10407537)

December 18, 2013

1 Individual steps of k-means

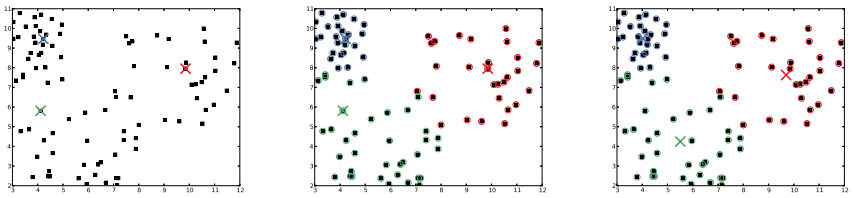
In the first step of the assignment we are focusing on implementing the k-means algorithm and check the results of these individual steps of the algorithm. Choosing the best parameter k always depends on the application. In this example where we have to cluster the datapoints which consist of three clusters, it is best to use $k = 3$ but this parameter varies for different applications. For example, image segmentation often needs around 10 (see for instance [1]) clusters to be computed. There is also a big trade off between number of clusters and the performance of the algorithm. We know that for a large value for k algorithm takes long time to converge. A really serious aspect of this algorithm is the relation between its random starting points in the state space and the solution that it gives. There is a big dependency and the algorithm can converge to a different final solutions depending on the starting points for the centers. Figure 1 depicts the results of the three steps of the algorithm with different values for k.

2 Running your implemented k-means

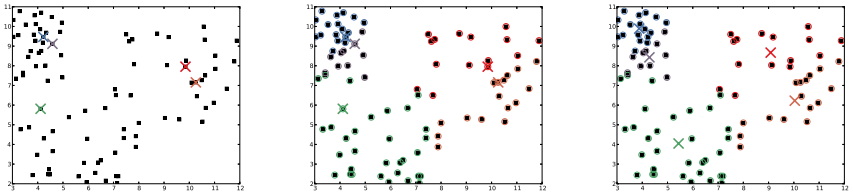
In this part of the assignment we apply the same algorithm but now use it iteratively. After the new assignment of the center points of the clusters we go to the second step again and we compute new centers. We continue the iterations until a maximum number of iterations is reached. Figure 2 depicts the results of the iterative method for 1, 5, 10 and 20 iterations. You can see in the figure that the centers are going towards the actual centers of the clusters. However, this is not always the case as the random first step can lead to different results.

Apart from reaching a maximum number of iterations, there is another way of stopping the algorithm, and this is done just by computing the distance between the previous centers and the new ones, once the distance has become not large enough to be important we can stop the algorithm. The following part of code is the one that we implemented for this reason.

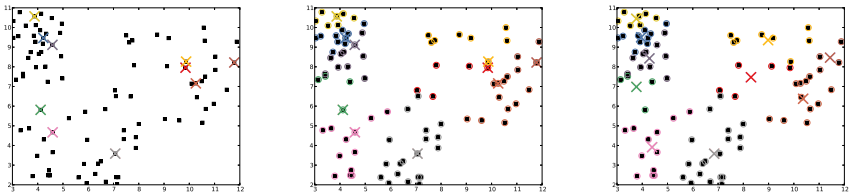
```
new_center_x = center_x/num
new_center_y = center_y/num
new_center_dist = math.sqrt(math.pow(new_center_x - means[i, 0], 2) +
```



(a) K=3

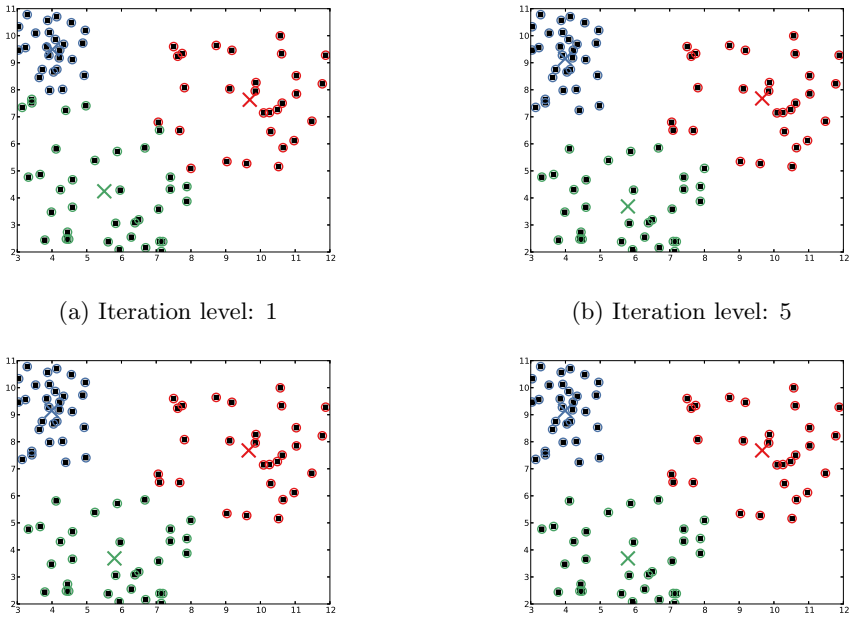


(b) K=5



(c) K=10

Figure 1: Illustration of the three first steps of the k-means algorithm for $k = 3, 5, 10$.



(a) Iteration level: 1

(b) Iteration level: 5

(c) Iteration level: 10

(d) Iteration level: 20

Figure 2: Illustration of the iterative k-means algorithm for different maximum iterations levels.

```
                    math.pow(new_center_y - means[i, 1], 2))
if new_center_dist > max_change:
   max_change = new_center_dist
   means[i,:] = [new_center_x, new_center_y]
if max_change < 0.001:
    break
```

3 Color-based image segmentation

Now, we apply the same algorithm on actual images. The main idea is the same but now we are dealing with a high dimensional space, so we expect this algorithm to be really slow. Thing that applies especially for large K values. We ran the k-means algorithm in the given image for $k = 3, 5, 10, 50, 100$. Figure 3 illustrates the resulting color segments of the image. We can see that for small values for k the image for example $K = 3$, each pixel is assigned a value of one of the three clusters, the resulting image has only three colors and different features in the picture can easily extracted. As the number of clusters increases the clustering is becoming more and more detailed. For $k = 100$ we can hardly see any difference between the clustered and the original image.

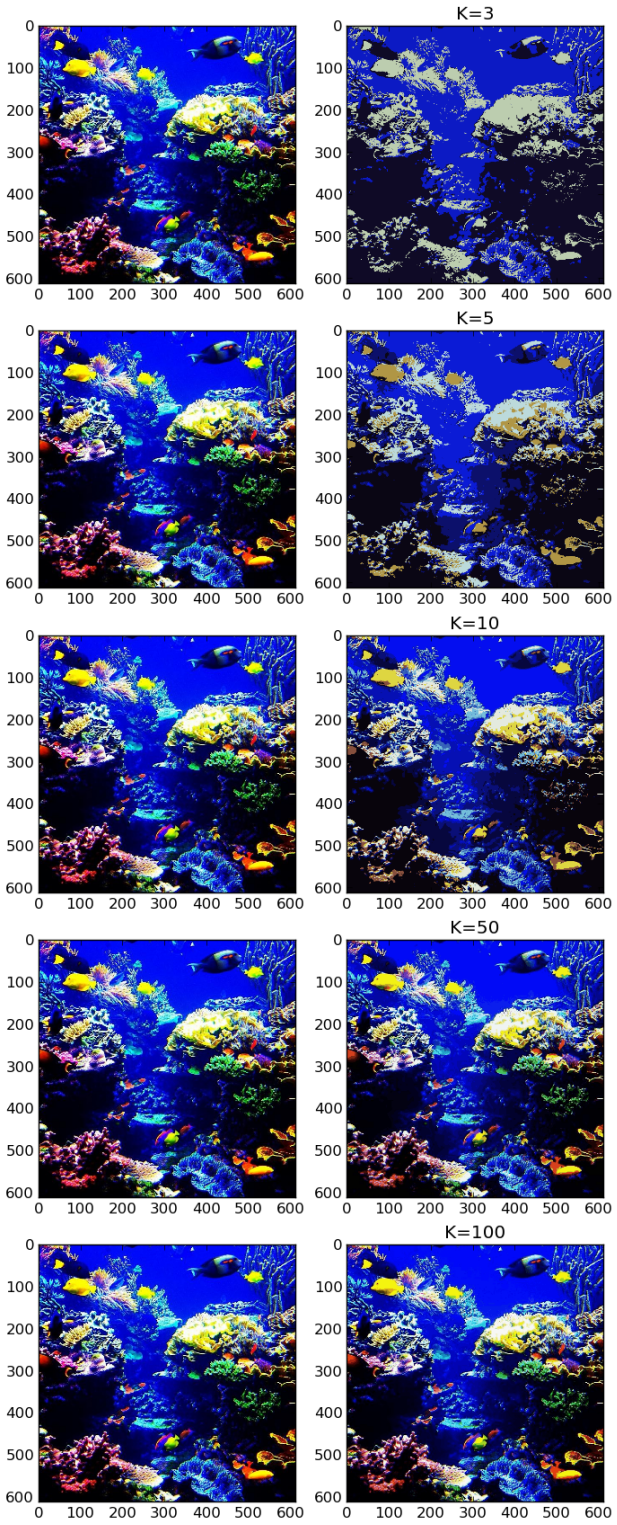


Figure 3: Illustration of k-means algorithm for the above coral image, for $k = 3, 5, 10, 50, 100$.

4 k-means, SIFT & Bag-of-Words: calculating storage for raw SIFT

Each SIFT vector has 128 elements. Each element is represented by an integer[1]. Assuming each integer consists of 2 bytes and there is no storage overhead for the vector, 1 SIFT vector needs 256 bytes of storage. To store all 50,000 vectors that belong to one image, 12,800,000 bytes $= 12.8 \cdot 10^6 B = 12.8$ MB ≈ 12.21 MiB is needed. For Facebook's daily addition of new images, one needs $12.8 \cdot 10^6 \cdot 350 \cdot 10^6$ B $= 4.48 \cdot 10^{15}$ B $= 4.48$ PB $= 4,480,000,000$ MB $= 4272460937.5$ MiB ≈ 3.98 PiB of storage.

5 Visualizing the words on the image, the contents of the words and the Bag-of-Words histograms

Visual words of five randomly selected images from the dataset have been plotted in Figure 4. Note that words tend to be focused on structures that reoccur often, such as lines. This can be seen in Figure 4a and 4b at the roof, and in Figure 4c and 4e at the 3D structures on the wall.

The patches of five randomly selected words are shown in Figure 5. The patches do look like each other, when you ignore the differences in hue (but not in intensity), rotation and scale.

Finally, the bag-of-words-histogram representation of the images of Figure 4 is presented in Figure 6.

6 Visualizing Bag-of-Words retrieval

See Figure 7 for the ranking.

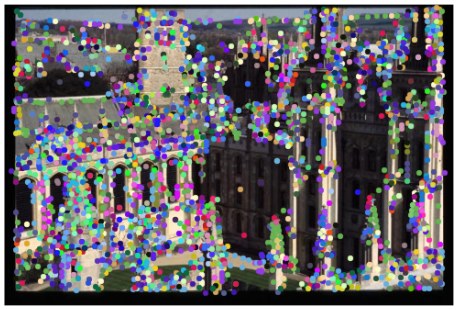
7 Evaluating Bag-of-Words histograms

Precision scores of the bag of words retrieval system can be found in Figure 8. Precision scores of the color histogram retrieval system for the same images can be found in table 1. The precision of the bag of words retrieval system seems to have an optimum for the codebook size. Smaller codebooks such as 5 and 10 seem to perform at least equally well in most cases as larger codebooks such as 500 and 1000. No strong conclusions can be made based on the current observations, as only a small number of queries have been examined, but this seems to tend towards overfitting; the data representation becomes too specific.

From the average precision scores, we might be able to conclude that the histogram distance performs better than the ℓ_2 distance, especially when few images need to be retrieved.

[1]The binary invoked to extract the SIFT vectors returns integers. However, the `read_sift_from_file(sift_path)` function interprets these as Python `float` objects. In the standard Python implementation, CPython, a `float` always consists of 64 bits, which is 8 bytes. This would result in quadrupling all final answers.

(a) `christ_church_000085.png`

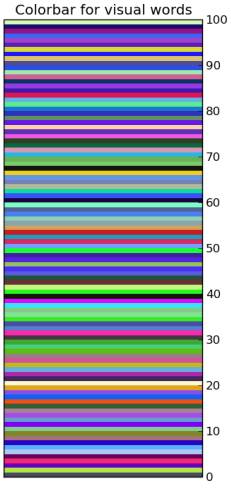


(b) `all_souls_000001png`

(c) `all_souls_000015png`

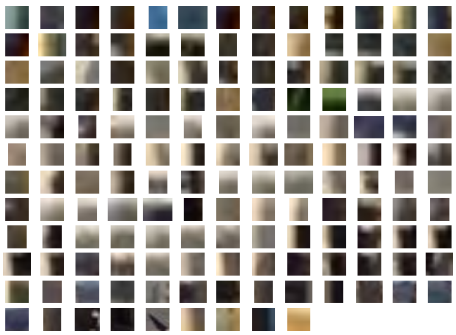
(d) `christ_church_000275png`

(e) `radcliffe_camera_000437png`

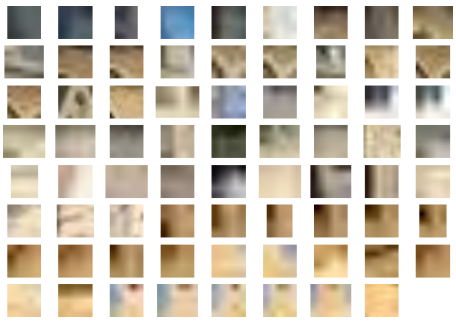


(f) Legend.

Figure 4: SIFT vectors that have been assigned to words, plotted on images. Figure 4f functions as a legend for each feature: dots of the same color represent the same SIFT vector. Its number can be looked up on this bar.



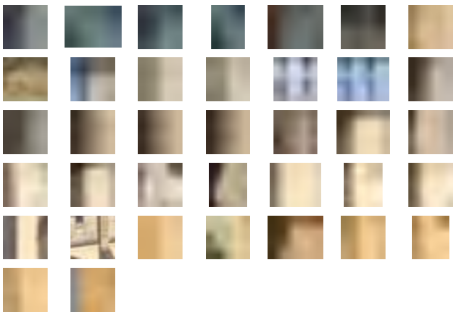
(a) Word 69



(b) Word 65

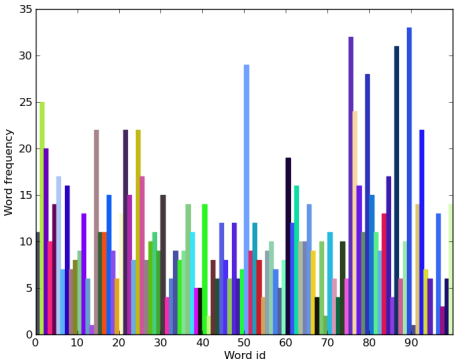
(c) Word 10

(d) Word 2

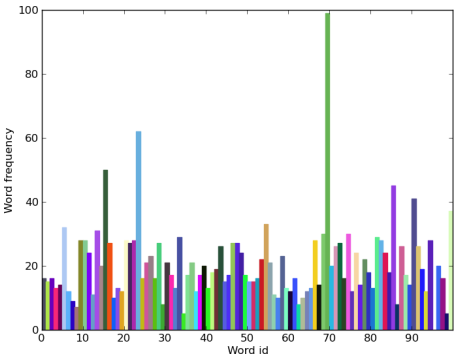


(e) Word 38

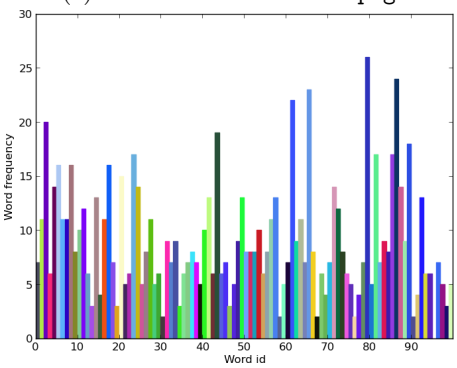
Figure 5: Patches for five randomly selected words, as they occur in the images of Figure 4.



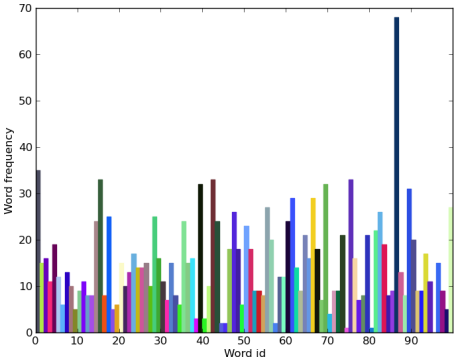
(a) `christ_church_000085.png`



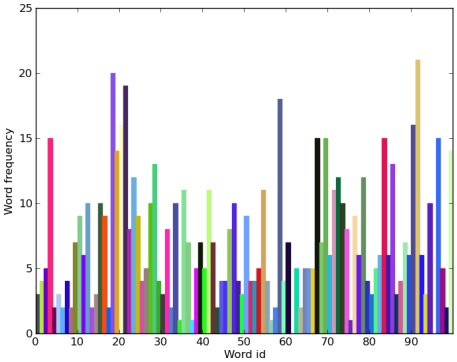
(b) `all_souls_000001.png`



(c) `all_souls_000015.png`

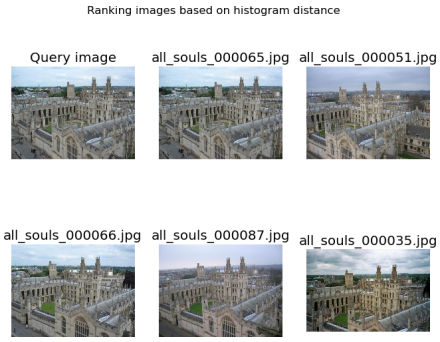


(d) `christ_church_000275.png`



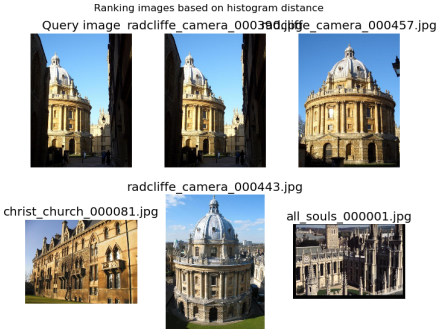
(e) `radcliffe_camera_000437.png`

Figure 6: Frequencies of visual words per image. Each bar is visualised in the corresponding SIFT vector color. Figure 4f shows these colors.

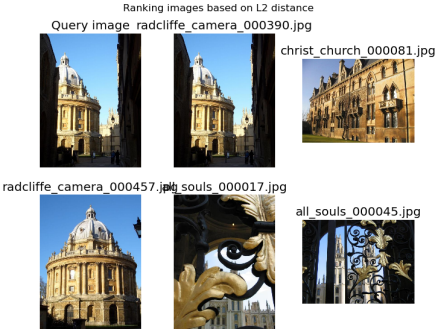


(a) Histogram distance, `all_souls_000065.png`.

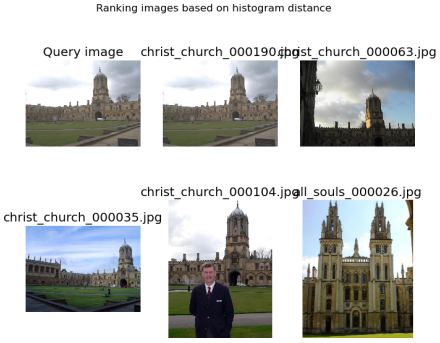
(b) ℓ_2 distance, `all_souls_000065.png`.



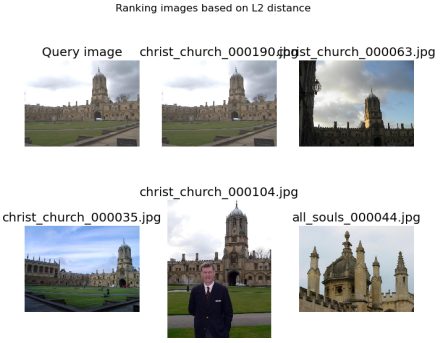
(c) Histogram distance, `radcliffe_camera_000390.png`.



(d) ℓ_2 distance, `radcliffe_camera_000390.png`.

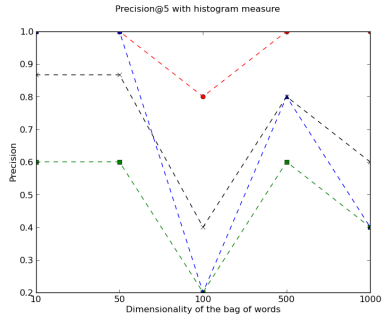


(e) Histogram distance, `christ_church_000190.png`.

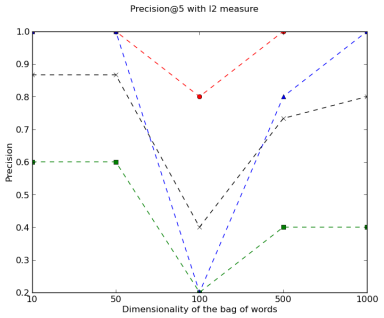


(f) ℓ_2 distance, `christ_church_000190.png`.

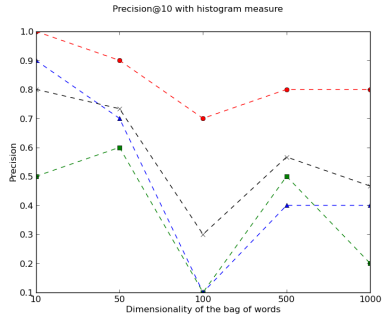
Figure 7: Ranking of retrieved images based on different distance measures between histograms of bags of words.



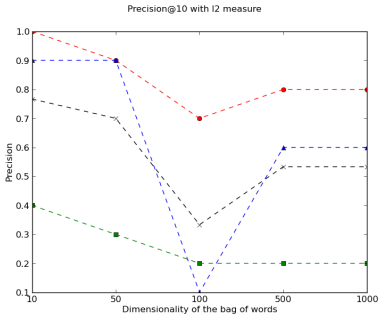
(a) Precision@5, histogram distance.



(b) Precision@5, ℓ_2 distance.



(c) Precision@10, histogram distance.



(d) Precision@10, ℓ_2 distance.

Figure 8: Precision@5 and precision@10 with histogram and ℓ_2 distance measures for the bag of words retrieval system. Results are shown for three different images, and their average scores. Red circles represent scores for `all_souls_000065`, green squares display values for `radcliffe_camera_000390`, blue triangles those of `christ_church_000190`, and black crosses stand for the average values.

We might also conclude that there are very few images like `christ_church_000190`, as neither measure returns high precision.

To propose a method to improve the accuracy of the system, we should investigate the current accuracy as well, which has not been done. A method to improve the precision might be to use a different feature descriptor than SIFT. One might want to look at histograms of oriented gradients, SURF or AGAST.

Comparing the bag of words retrieval system with the color histogram retrieval system, we can observe that only in a few cases the color histogram system outperforms the bag of words system. In general, the bag of words system outperforms the color histogram system. This is not that strange; the bag of words system is based on a more advanced feature descriptor method than color histograms.

References

[1] Georgios Methenitis, Patrick M. de Kok, Sander Nugteren, and Arnoud Visser. Orientation finding using a grid based visual compass. In *25th*

all_souls_000065	0.8
radcliffe_camera_000390	0.4
christ_church_000190	0.2
Average	0.47

(a) Precision@5, histogram distance.

all_souls_000065	0.6
radcliffe_camera_000390	0.0
christ_church_000190	0.0
Average	0.2

(b) Precision@5, ℓ_2 distance.

all_souls_000065	0.5
radcliffe_camera_000390	0.3
christ_church_000190	0.1
Average	0.3

(c) Precision@10, histogram distance.

all_souls_000065	0.4
radcliffe_camera_000390	0.1
christ_church_000190	0.2
Average	0.23

(d) Precision@10, ℓ_2 distance.

Table 1: Precision@5 and precision@10 with histogram and ℓ_2 distance measures for the color histogram retrieval system. Results are shown for three different images, and their average scores.

Belgian-Netherlands Conference on Artificial Intelligence (BNAIC 2013), pages 128–135, November 2013.