Computer Vision Exercise II: Report

Alexander Cech
Student of Visual Computing
Vienna University of Technology
e08900070@student.tuwien.ac.at

Hamed Jafari-Sahamieh
Student of Visual Computing
Vienna University of Technology
e09826767@student.tuwien.ac.at

Patrick Link
Student of Visual Computing
Vienna University of Technology
e11728332@student.tuwien.ac.at

1 Assignment 4: Image Stitching

The goal of the assignment is to combine multiple overlapping photos of a scene into an (ideally seamless) panorama. In order to do that, first corresponding points in the overlap region between images have to be found. From these correlations a homography can be calculated to project one of the images onto the other. Finally the homographies between neighbouring images are combined to a global mapping, so each image can be projected onto its final place in the panorama image. To test our implementation, besides the two provided image sets, we used a set of photos we took ourselves, and – as an example for artificially generated images – screenshots from the computer game Doom. These additional image sets are shown in Figure 1.

Figure 1: Additional image sets used for experimentation.

1.1 SIFT Interest Point Detection

In the first step we used the function `vl_sift` to run the Scale Invariant Feature Transform (SIFT) [2] algorithm for key point detection. As we did not need the key descriptors at this stage, we only acquired the position of each key point, along with its scale and orientation. SIFT uses Difference of Gaussians (DoG) to approximate the Laplacian of Gaussians (LoG): to examine different scales, the SIFT algorithm creates a minimum of five different Gaussian images per octave; the difference of these images (DoGs) are then used for maxima detection, similar to the blob detection algorithm from the previous assignment.

To find the orientation of the key points, depending on the scale, a region around the key point center is chosen. The gradient magnitudes and orientations are calculated for each pixel in this region, and the result is placed in a histogram consisting of 36 bins (10° per bin). The largest bin of the histogram is the orientation of the key point; other bins very close to the maximum are converted into new key points (same position and scale, but with a different orientation). The key point orientation is important, because further calculations (descriptors) will be relative to this orientation, which ensures orientation invariance for matching key points.

The function `vl_plotframe` illustrates the results. A circle's radius corresponds to the scale of the associated key point and the line is its orientation as visualized in Figure 2.

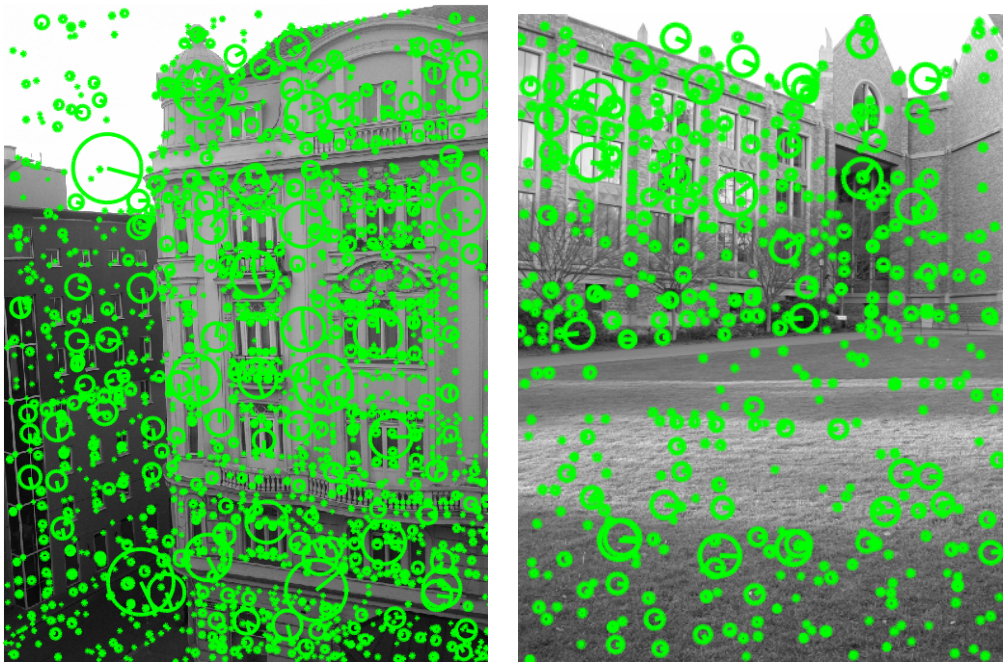


Figure 2: SIFT key point detection: scale and orientation of key points.

1.2 Interest Point Matching and Image Registration

Similar to the previous task, first we extracted all key points via SIFT, this time along with the associated descriptor of each key point. The descriptors are an array of 128 numbers constructed as follows: First the 16x16 window around the key point is broken down into smaller 4x4 windows. Within each of the 16 windows the gradients are calculated and subsequently put into an 8 bin histogram (45° each). The magnitude contributed by each gradient also depends on the distance from the key point (using Gaussian weights).

Each of the 128 resulting numbers is normalized and reduced by the key point's orientation (to ensure orientation invariance) and subsequently clamped at 0.2 (to ensure illumination invariance) before being normalized to a unit vector again.

`vl_ubcmatch` is used as a first, basic, matching step. For each descriptor in the first image, `vl_ubcmatch` attempts to find the closest descriptor in the target image using the L2 norm[1] of the difference between the two. This basic algorithm may return many false positive matches though. The result is illustrated in the top row of Figure 3.

To eliminate potential false positive matches, a RANdom SAmple Consensus (RANSAC) [1] scheme is applied: Basically, a homography is estimated by using 4 random sample matches out of all matches found by `vl_ubcmatch`, which is then applied to all other key points via `transformPointsforward`. The number of inliers, i.e., those matches having their Euclidean distance smaller than a certain threshold T, is determined and remembered. This procedure is repeated N times, and finally only the inliers of the best homography found (i.e., the one yielding the highest number of inliers) are used to estimate the final homography. Figure 3 shows the remaining (inlier) matches after running RANSAC in the middle row, and the false positive matches eliminated in the bottom row.

After applying the above steps the images are considered well aligned. Figure 4 shows absolute differences and overlays of pairs of matched images, using the estimated homography.

As described above, in a key point descriptor the gradient orientation of each pixel is adjusted to the key point's dominant orientation before being added to the associated histogram. Therefore the results should be invariant to changes in image rotation. Additionally the SIFT algorithm is also scale invariant as described in the first section. That this is indeed the case can be seen in Figure 5, where one of the images has been rescaled and rotated.

[1] Least squares, Euclidean norm

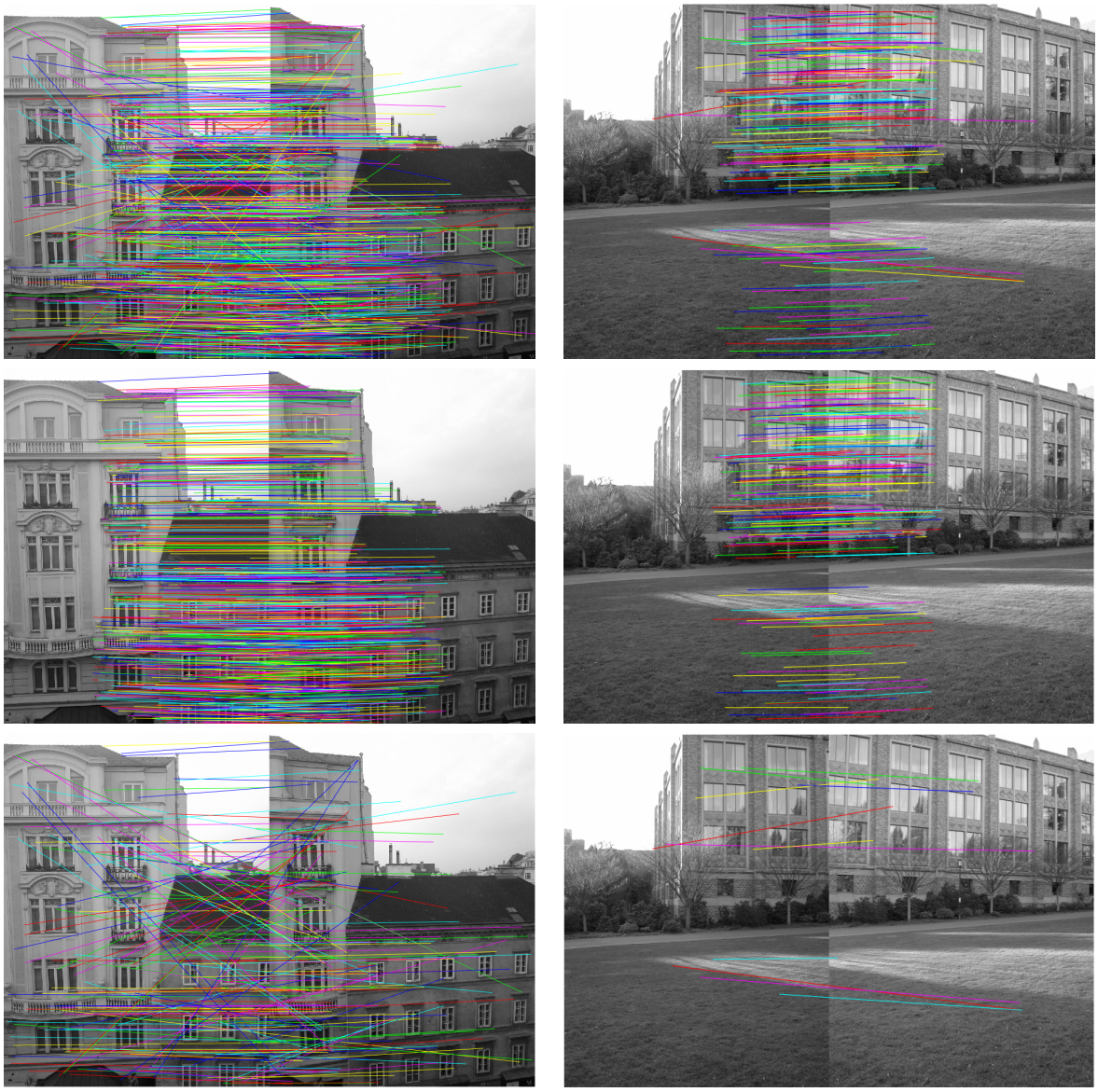


Figure 3: Top: Initial matchings obtained by `vl_ubcmatch`. Middle: Correct matches (inliers determined by RANSAC). Bottom: False matches (outliers removed by RANSAC).

Figure 4: Difference (left side, dark areas) and overlay (right) view of aligned images.

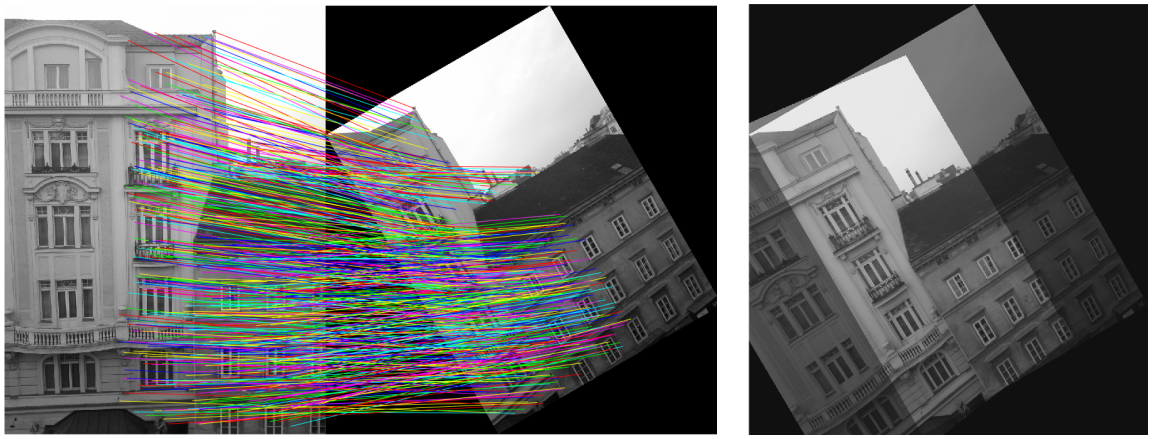


Figure 5: Post RANSAC matches after rotating and rescaling and the aligned result.

1.3 Image Stitching

To stitch images together for the purpose of creating a panorama view, first the homography is estimated between each pair of adjacent images. One image is selected as reference image (usually the one in the centre of the panorama) and, utilizing the homographies estimated before, the homography of each image to the reference image is calculated. From this complete set of homographies, the final size of the panorama is calculated, and each image then is projected onto its correct place within the panorama. The results of independently projecting the images is shown in Figure 6.

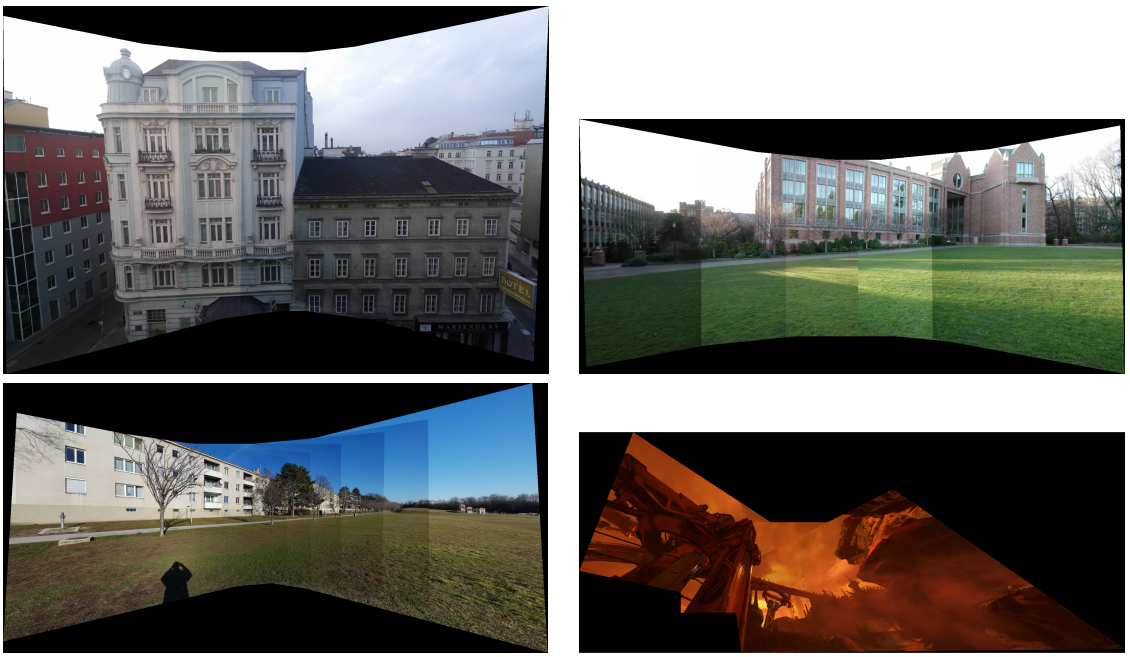


Figure 6: Resulting panorama images without blending.

However, just overlaying the images results in clearly visible seams, due to different brightness. In order to create a seamless panorama, a feathering technique is used: the basic idea is, that the centre values of each image are more important than its border values. For this purpose, we calculated a normalised distance-to-border map for each image, using MATLAB's `bwdist` function (a kind of α-channel), representing the relative importance of each image pixel. When projecting the images to the panorama, this mask is also projected with the same homography, and the final color for each pixel is calculated as a weighted sum of the image contributions, the weight being the relation of each image's alpha value to the sum of alpha values for this pixel:

$$O(x,y) = \frac{\sum_{i=1}^{n} (R_i, G_i, B_i)\alpha_i}{\sum_{i=1}^{n} \alpha_i}$$

The result of this procedure is that the overlapping regions are smoothly blended between the contributing images, as demonstrated in Figure 7.

Although the generated panorama images look pretty convincing, upon a closer look one can find different problem spots. These result mainly from imperfect alignment and from image areas of the same location which are different in the separate images. Examples for sources for the latter problem are shadows present in one image, but not in another, or passengers (or other moving objects) that only are captured in some of the images, resulting in "ghosts". Both of these problems are most visible in a blended panorama. In unblended panoramas the most visible artefacts besides the seams are misalignments, which result in jagged areas. Representatives of each of those problems are shown in Figure 8.

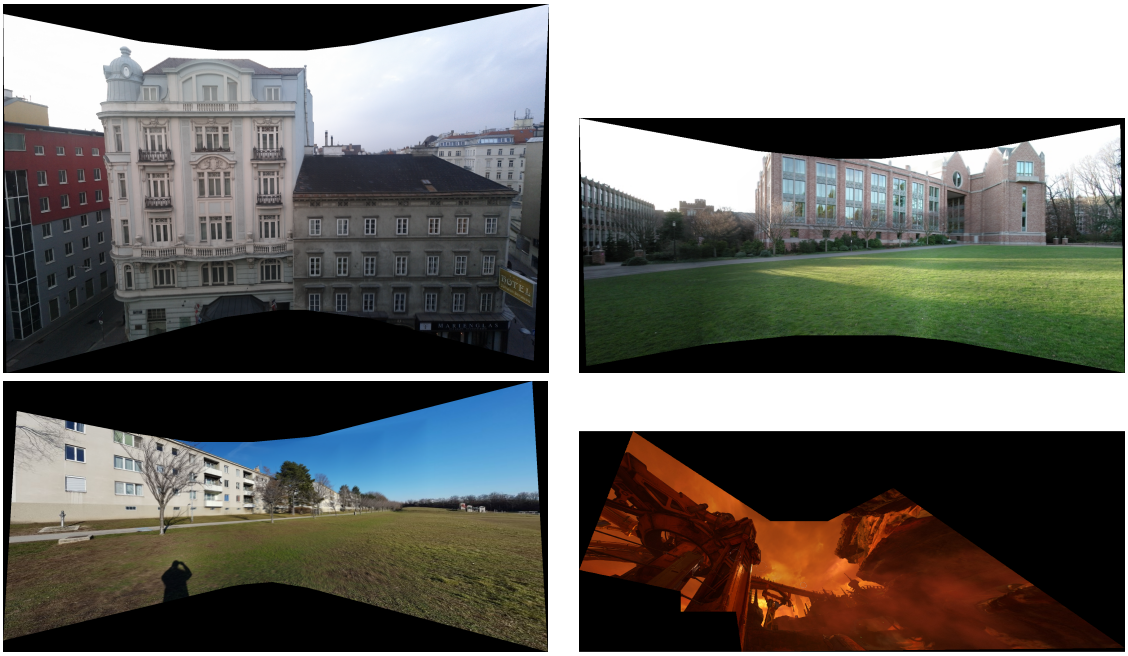


Figure 7: Resulting panorama images with feathering.

Figure 8: Problematic areas (left column: unblended, right: feathered). Top: Imperfect alignment, resulting in double-images. Middle: Ghosting of passengers who moved between taking the separate images. Bottom: Jagged connections in the unblended panorama.

2 Assignment 5: Scene Recognition with Bag of Visual Words

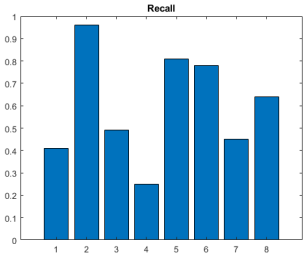
The goal of this section is to map any test image to 8 pre-defined scenes described by a training set of images. For this purpose, we try to create a vocabulary of visual words. The goal of a visual word is to be the most expressive descriptor for a type of interest point. As the first step, we use `vl_dsift` to get around 100 descriptors from each image on the training collection. We perform k-means clustering on the list of all resulting descriptors to get our visual words. Each visual word therefore is the centroid of a cluster of 128 numbers long SIFT descriptors.

In our tests we utilized both 50 and 75 clusters/visual words.

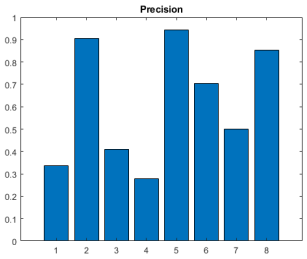
Once the vocabulary is complete, we need to create a map between the existing training images and our visual words. First we use `vl_dsift` to extract the key points of each training image, however this time with a smaller step size. For the exact step size 2 was chosen as the results did not improve with the size of 1. Next we use knnsearch to find the nearest visual word in vocabulary for each SIFT descriptor of the image. Finally we create a (normalized) histogram for each image representing the occurrence count of each visual word in the image. The collection of all histograms(training) and the associated image classes (group) is returned by the function `BuildKNN`.

The next step is to test images against our model as an attempt to recognize the scene correctly. Again, we use `vl_dsift` and create a histogram as described above. As the function `knnclassify` is removed in MATLAB2018 we used the combination of functions `fitcknn` and `predict` to guess the correct scene. The result of the prediction compared to the actual class of the image is reflected by the confusion matrix.

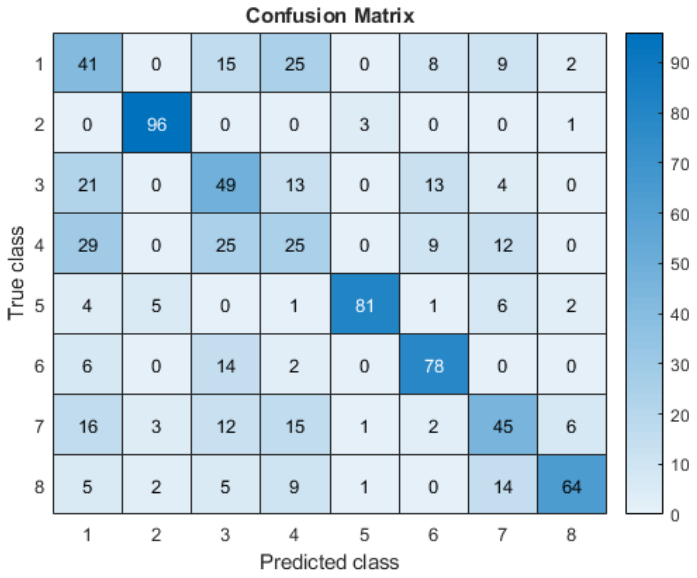
Figures 9 to 13 visualize the precision, recall and the confusion matrix for different sizes of vocabulary, step parameter and number of neighbors. Precision is the number of correclty classified images divided by the number of images classified to this group, it measures how often a image classified as a class has actually this class. Recall is the number of correctly classified images divided by the number of images of this class. It measures how many images of a class where correctly classified.



(a) Recall



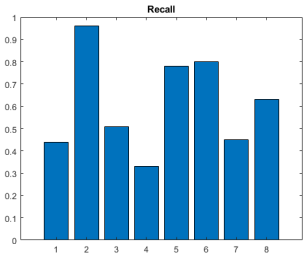
(b) Precision



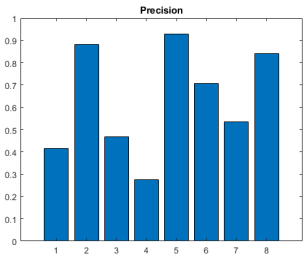
(c) Confusion Matrix

Figure 9: Results for numClusters = 50, numNeighbors = 3 and step size of 2. Success rate is 59.9%.

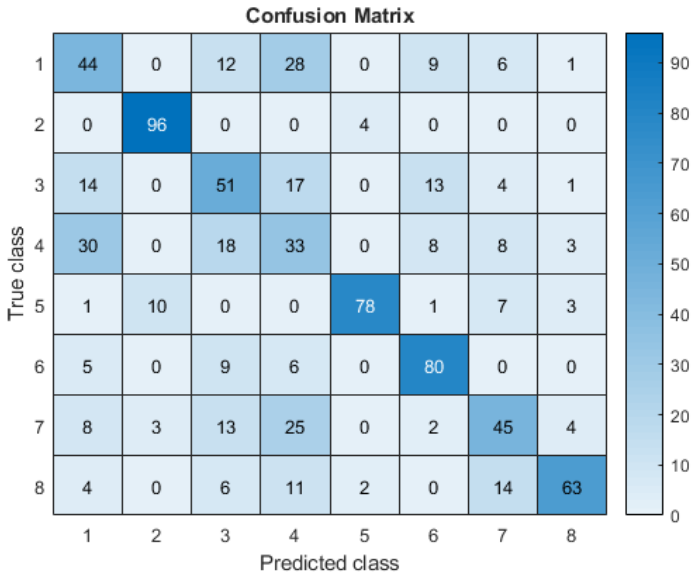
As illustrated by the diagrams, the success rates of the classes mountain, forest and office is quite high. The classes bedroom, kitchen and living room have a high rate of confusion with each other. Especially the living room's bad detection rate is mostly due to the other 2 categories. There is a certain confusion between mountain and forest, however that can be seen as expected. We can also see that the recall of class bedroom is very bad, as many of these images are classified as livingroom. On the other hand the precision of the class bedroom is quite high in the last run, because almost only bedrooms get classified as bedrooms, but no other of these confusing classes. On the other hand the class livingroom has a very low precision, as most of these room classes get classified as living room. The class street lies in between these two groups of easy and hard classsification examples, when it gets confused for an other



(a) Recall



(b) Precision

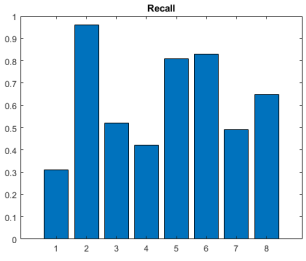


(c) Confusion Matrix

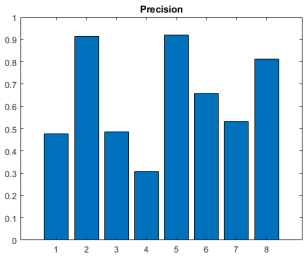
Figure 10: Results for numClusters = 75, numNeighbors = 3 and step size of 2. Success rate is 61.3%.

class, it's often a mandmade scene, but the precision is relatively high.

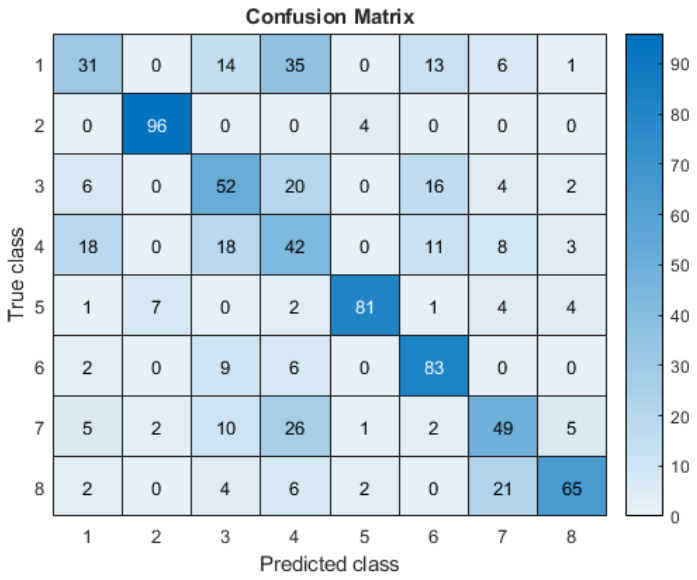
We also tested the bag of words classifier on own images. We used two pictures of a forest, one picture of a kitchen, one of a lliving room, 5 pictures of mountains and 3 pictures of a street. The pictures were scaled down to match the resolution of the training set. In figure 13 the results are shown. The complete overall success rate was 58.3%. Strangely the images of the class forest, livingroom and kitchen were correctly classified. Two mountain images were classified as forests, but there is a lot of forest in the mountain pictures. The street images were classified as livingroom, kitchen and store, even if they don't resemble each other for a human. The results on the test set and our own images lead us to believe that we can expect a classification rate of about 60% on new unseen pictures. In fiigure 14 some of our images are shown.



(a) Recall

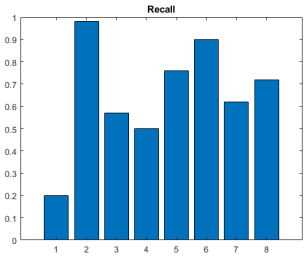


(b) Precision

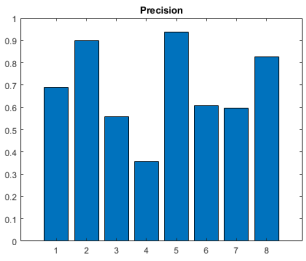


(c) Confusion Matrix

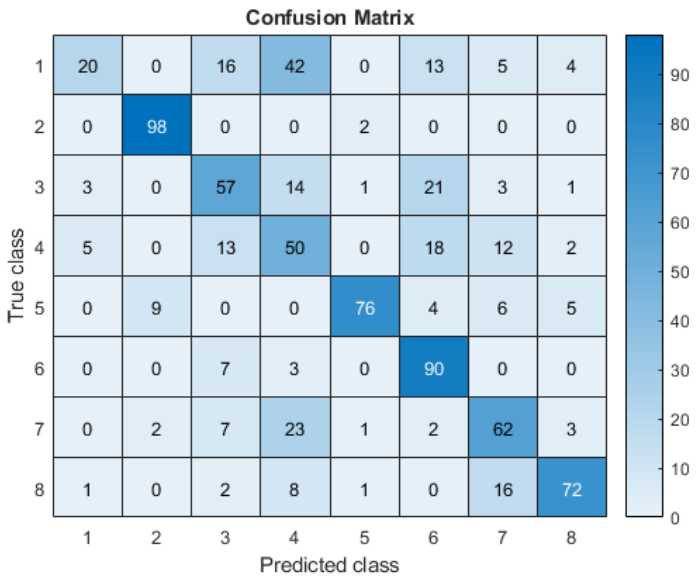
Figure 11: Results for numClusters = 75, numNeighbors = 5 and step size of 2. Success rate is 62.4%.



(a) Recall

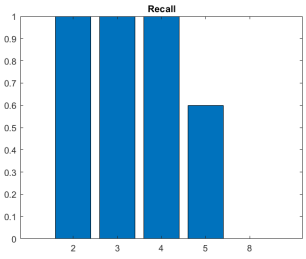


(b) Precision

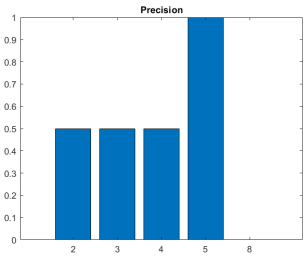


(c) Confusion Matrix

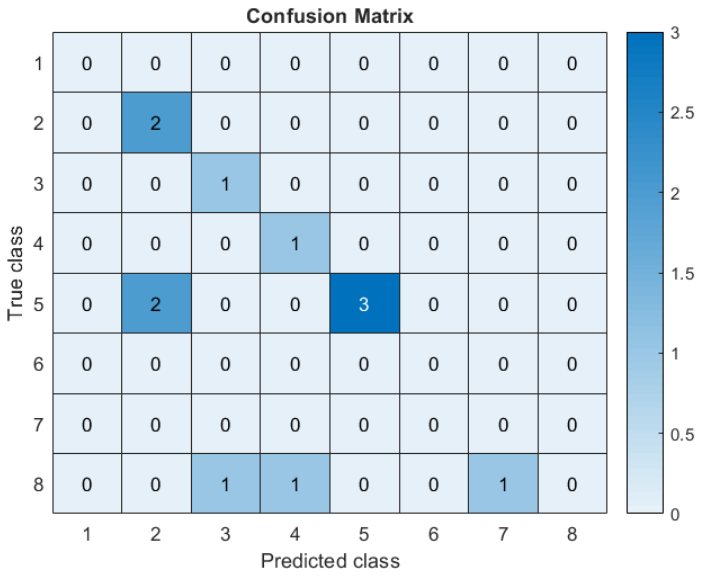
Figure 12: Results for numClusters = 75, numNeighbors = 21 and step size of 2. Success rate is 65.6%.



(a) Recall



(b) Precision



(c) Confusion Matrix

Figure 13: Results for own images. numClusters = 75, numNeighbors = 21 and step size of 2. Success rate is 58.3%.

(a) Street

(b) Mountain

(c) Kitchen

Figure 14: Some of the test images

References

[1] Fischler, Martin A. and Bolles, Robert C. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", *Communications of the ACM*, 24(6):381–395, 1981.

[2] Lowe, David G. "Distinctive Image Features from Scale-Invariant Keypoints", *Journal of Computer Vision*, 60(2):91–110, 2004.