Universitat Politècnica de Catalunya
Universitat de Barcelona
Universitat Rovira i Virgili

Master in Artificial Intelligence

Computational Vision

Retrieval of Images Based on Texture

Authors:
Alejandro Suárez Hernández
Johannes Heidecke

January 2017

Contents

List of Figures

1 Introduction

For this report we evaluate the capabilities of texture-features extracted with the Leung-Malik (LM) filter bank of finding images similar to a given one. In the following sections we will extract descriptors based on texture filters and then use a knn-search algorithm to retrieve the most fitting images.

2 Leung-Malik Filter Bank

We use the Matlab code provided by the Visual Geometry Group of Oxford University to create the LM filter bank[1]. The bank consists of 48 different filters at different scales and orientations: 36 first and second derivatives of Gaussians at 6 different orientations and 3 scales, 8 Laplacian of Gaussian (LOG) filters, and 4 Gaussian filters. Figure 1 shows the filters visualized as heat maps.

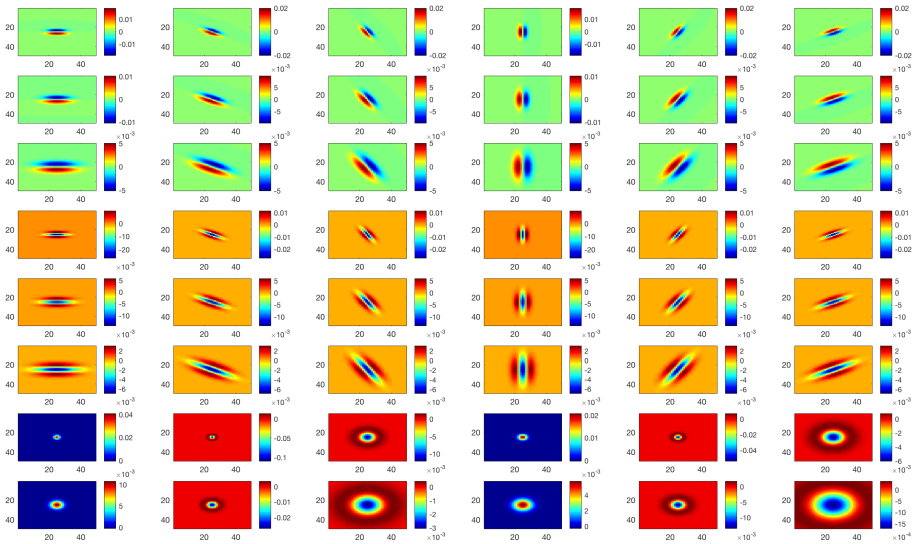


Figure 1: Visualization of LM filter bank

3 Texture Descriptors

To obtain expressive descriptors that describe the texture of an image, we use the responses of the image to the convolutional filters of the LM filter bank.

Figure 3 shows the response of the image `forest_9` on all 48 filters (to compare filters and responses, also see figure 1). In figure 2 we show the original image in RGB colors. Figure 3 depicts the responses of the same image after being converted to grayscale to all 48 filters as a heatmap with blue corresponding to low responses and red corresponding to high responses.

Figure 4 normalizes the visualization of responses so they all lie on the same scale between the global minimum of -0.2982 and maximum of 0.1799. It can be observed that the filters of vertical first gaussian derivatives get a strong result on this image, most

[1] `http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html`

Figure 2: Example image `forest_9`

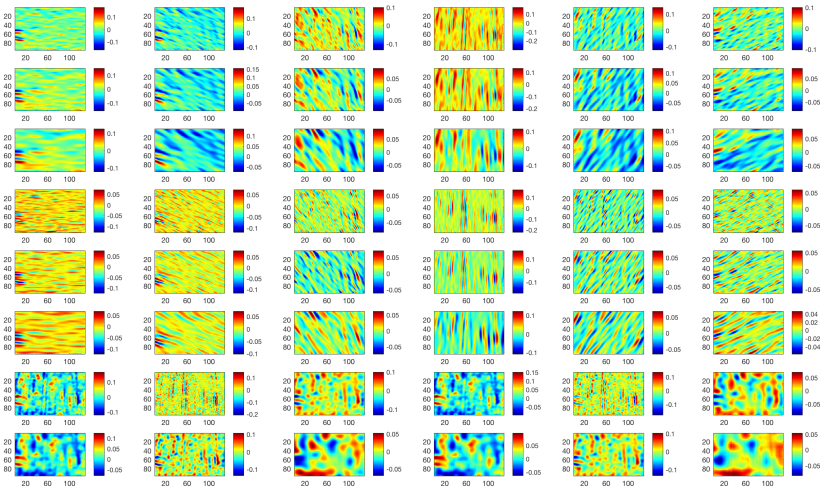


Figure 3: Responses of `forest_9` to the LM filter bank

probably caused by the edges of the vertical tree trunks. The responses for filters with a horizontal orientation on the other hand are much lower.

As comparison to the forest scene, responses to `buildings_2` (figure 5) can be seen in figure 6. Here, the slight diagonal filters yield a relatively high response. The responses clearly differ between the image of the forest and the image of a building. The first derivatives of gaussians yield a high response along the edges of the building.

To evaluate the difference of responses between the different classes of images more quantitatively, we examined the boxplots of the responses showing mean, standard deviation and range of values in figure 7. Subfigure (a) shows boxplots for all 48 filter's responses for the `forest_9` and subfigure (b) the corresponding responses of `buildings_2`. It can be seen easily, that the responses have a very different distribution of values for the two images.

In figure 8 we show the differences of mean responses and standard deivation of the responses. Again, we can observe a quite unique distribution for both images.

4 Class Feature Matrices

When applying the convolutions of the LM filter bank, responses for the filters are usually different in different parts of an image. In order to describe an image's texture with a small but expressive number of descriptors, we need to aggregate the responses of the image for

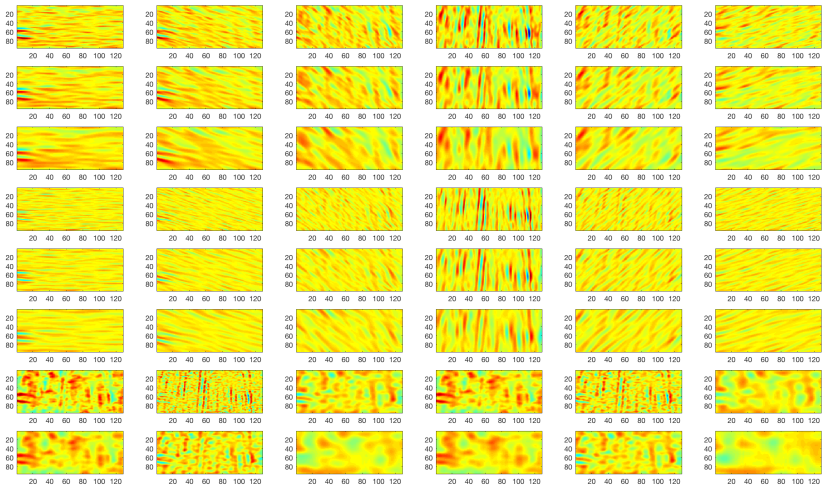


Figure 4: Normalized responses of `forest_9` to the LM filter bank.

Figure 5: Example image `buildings_2`

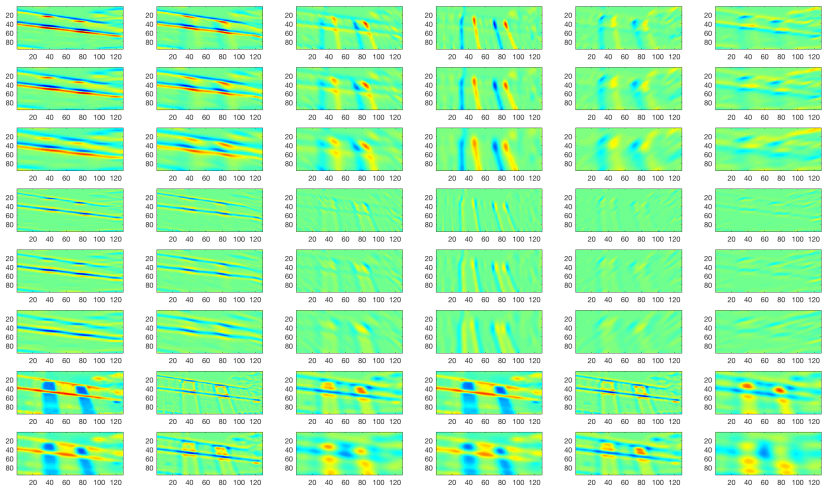
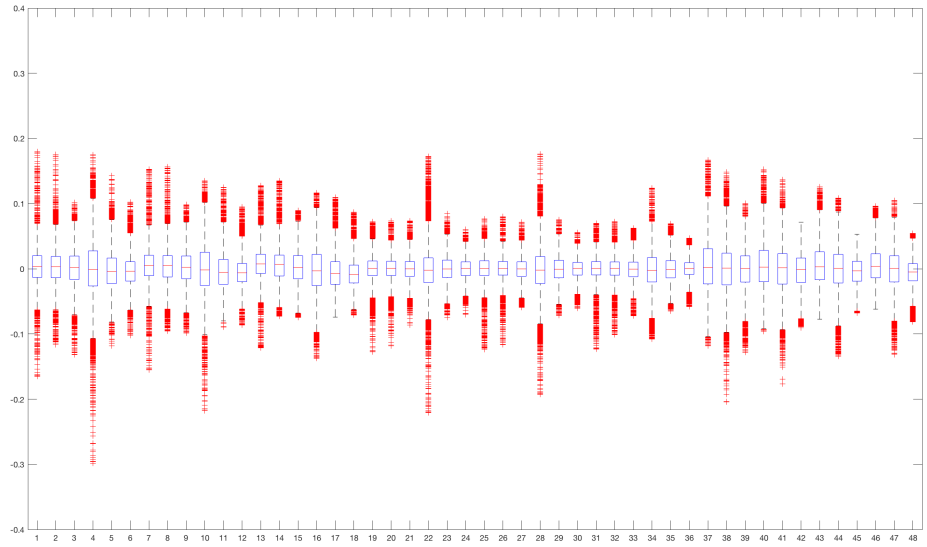
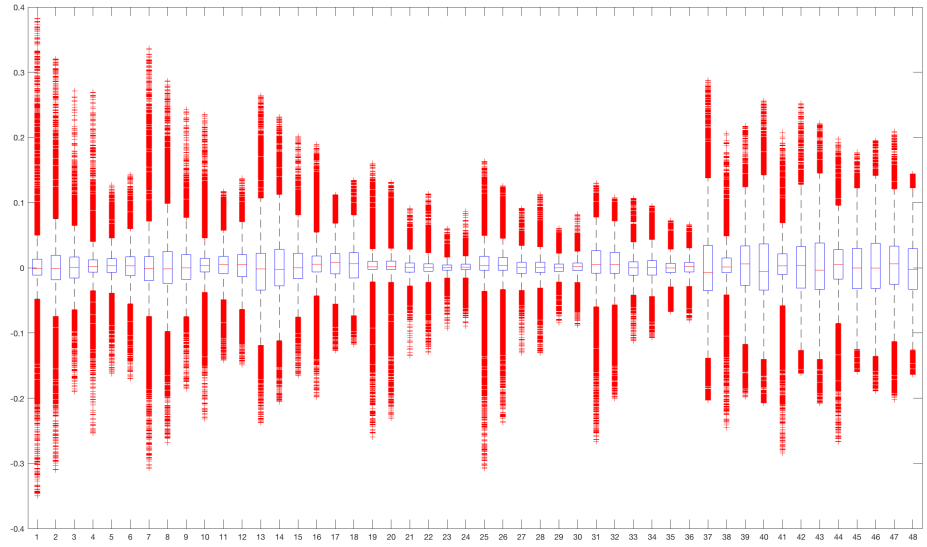


Figure 6: Normalized responses of `buildings_2` to the LM filter bank

each filter into single values. For this we can calculate descriptive metrics such as mean, standard deviation, or median. For example, the mean response for the first filter will be



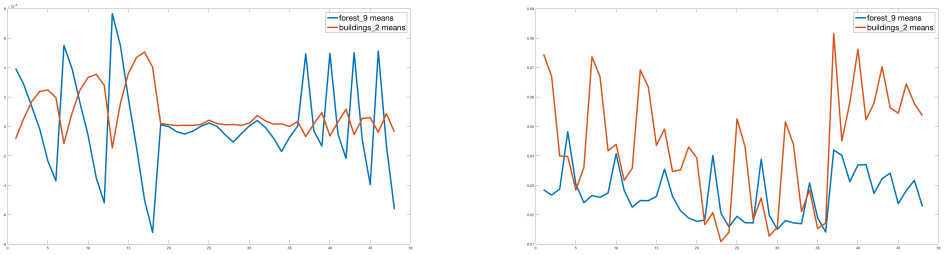
(a) Responses for `forest_9`



(b) Responses for `buildings_2`

Figure 7: Boxplots of responses to LM filter bank

one feature, the mean response for the second filter another one, and so on. We construct a feature matrix for each class (forest, buildings, and sunset) of images with each row corresponding to an image and each column corresponding to the aggregated response of this image to a filter of the LM filter bank. In the following sections we explored the following aggregations: mean, standard deviation, minimum, maximum, and median. As we show in section 5, not all aggregations lead to the same separability of the different classes of images. Our experiments showed, for example, that the class of sunset images can be distinguished best from other classes when looking comparing the standard deviations of results.



(a) Mean responses (b) Standard deviation of responses

Figure 8: Comparison of mean and standard deviation

5 Visualizing Class Feature Differences

For each image we obtain a vector of aggregated responses for each filter of the LM filter bank. In this section we show some examples of how well these descriptors are able to be used to separate the classes. Figure 9 plots the mean values of filter 41 (x-axis) and filter 25 (y-axis) for all 90 images. Green dots correspond to forest images, blue dots to buildings and red dots to sunsets (this color code is used through the entire report). The classes are clearly not linearly separable based on these two features alone. There also is no obvious and elegant non-linear separation.

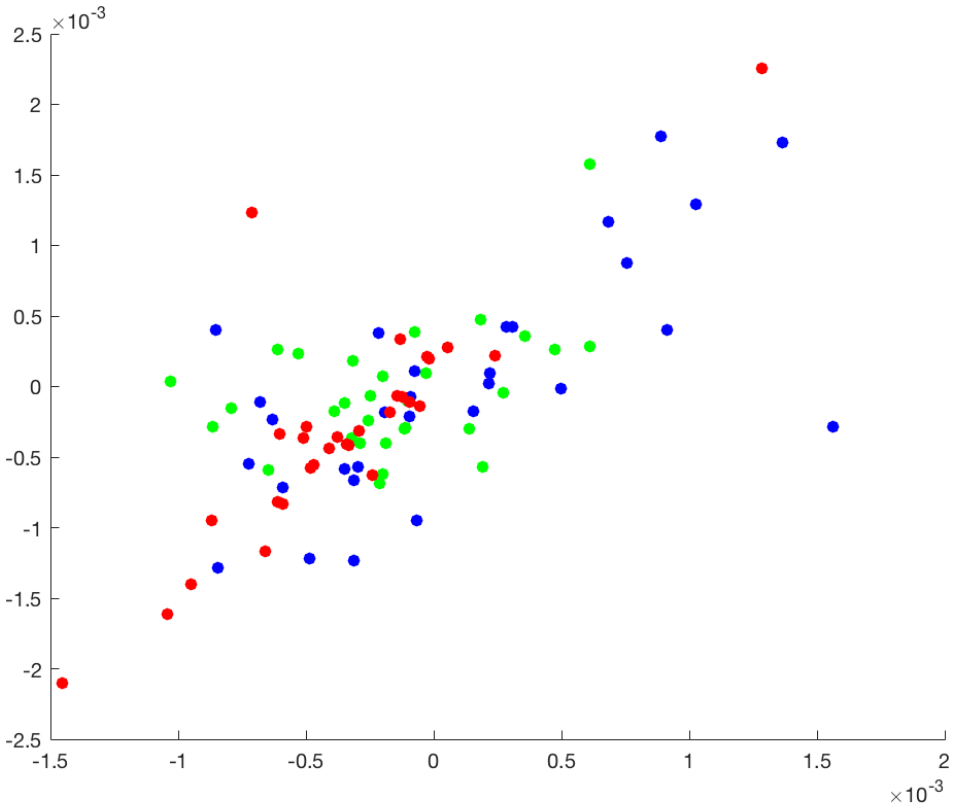


Figure 9: Mean responses to filter 41 and filter 25

Figure 10 compares the standard deviations of the responses to the same filters, 41 on the x-axis and 25 on the y-axis. We can see here, that the classes are much better separable.

Especially the sunset class tends to have a higher standard deviation of the filter 25 response and can - ignoring some outliers - be separated linearly. The building and forest class lie more closely together, but even here a non-linear separation can be assumed. For these two filters, standard deviation seems to be a better aggregation to separate the different classes.

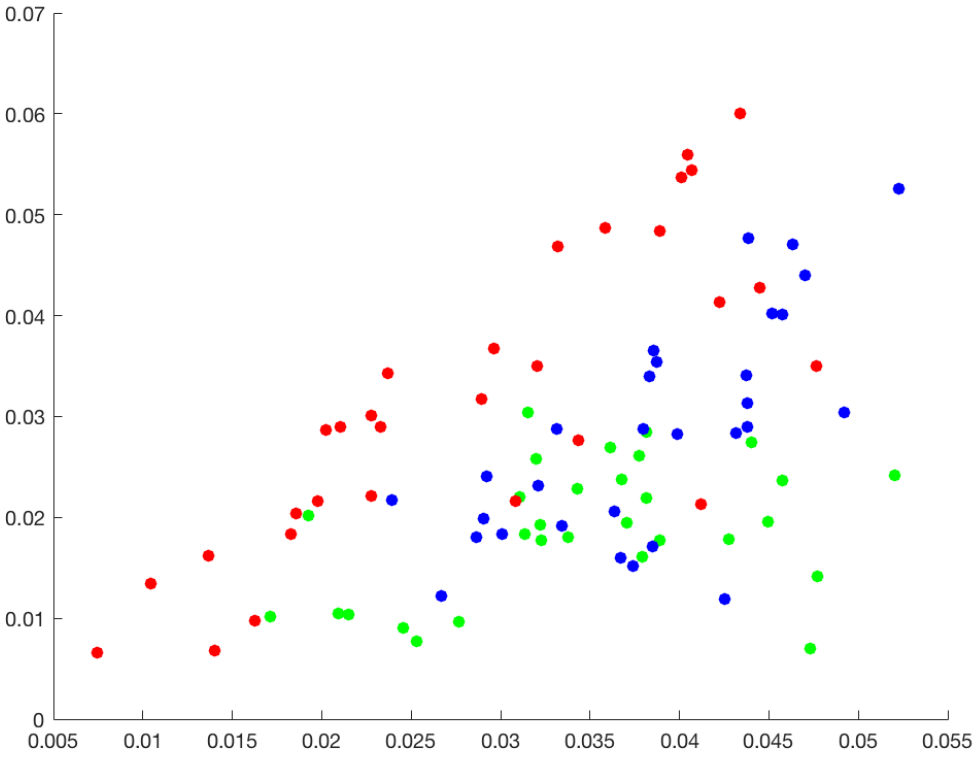


Figure 10: Standard deviation of responses to filter 41 and filter 25

Another good result can be seen when comparing the mean of responses to filter 21 (x-axis) to the standard deviation of responses to the same filter (y-axis) in figure 11. Like in the previous example, the sunset class is more distinguished from the two other classes.

6 KNN Search for Similar Images

The objective of this report is to evaluate the capabilities of texture features for retrieving images that are most similar to a queried image. For this, we use a knn (k nearest neighbors) search algorithm that returns the 9 images whose features are most similar to the given image, using euclidean distance to quantify similarity.

In figure 12 we show the results of retrieving 9 similar images for both `forest_9` and `building_5` when using only the means of responses as features. The retrieved images are sorted by similarity from top left to bottom right.

We can see that the algorithm does succeed with retrieving some similar images, but also fails in some cases. For the forest image the result does contain a majority of other forest images, but the most similar image falsely is an image of a skyline and there is also an image of a sunset. The image `building_5` is a bit more ambiguous, since it depicts both buildings but also grass and bushes that might have a similar texture to forest images. The results support this suspicion, consisting of a mix of forest and building images.

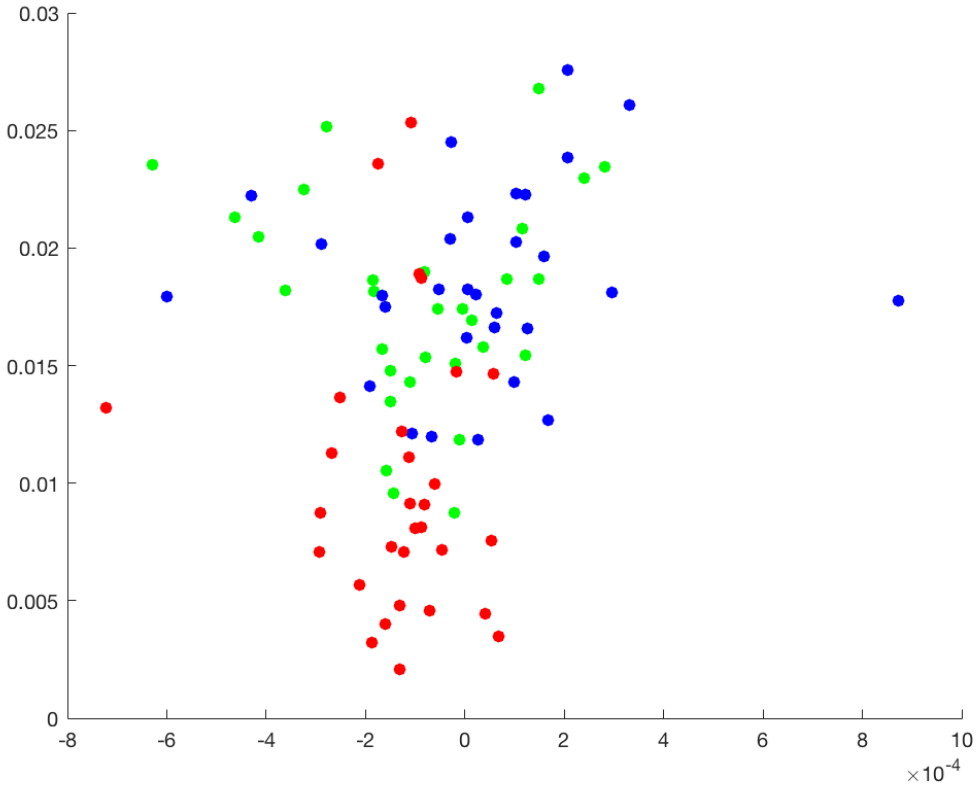
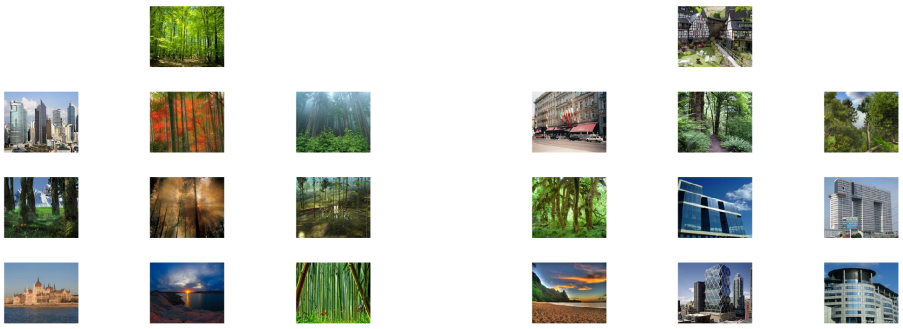


Figure 11: Mean and standard deviation of responses to filter 21

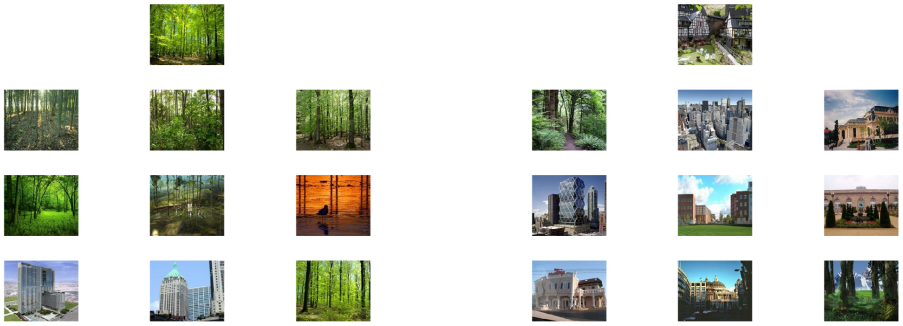


(a) `forest_9` (b) `buildings_5`

Figure 12: Example KNN results for mean based features

In figure 13 we see the results for using standard deviation of filter responses as features. The results clearly improve over the ones obtained by using means as aggregation. Especially for the forest example, the top images are now all very similar forest landscapes. For the building example we still get a mix of forest and building images, but the sunset example disappeared.

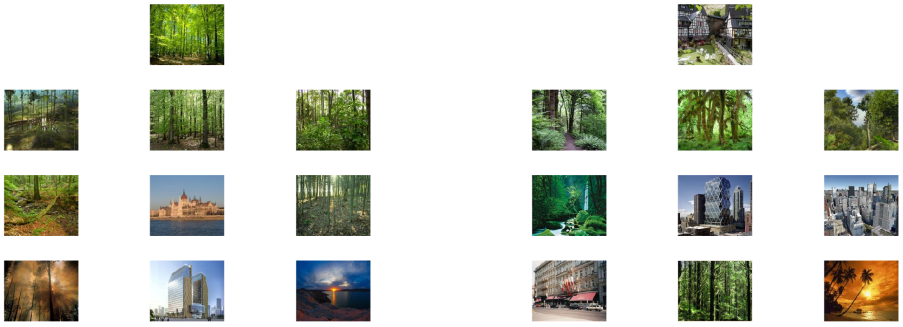
Figure 14 shows the results when using all aggregations methods at once and building a big feature matrix consisting of means, standard deviations, minima, maxima and medians. The results seem to decrease in quality when using all features combined - possibly due to the "curse of dimensionality" when using distance functions in high dimensional spaces.



(a) `forest_9` (b) `buildings_5`

Figure 13: Example KNN results for standard deviation based features

Some sort of feature selection would be necessary here to improve the results again. This, however, was beyond the scope of this report.



(a) `forest_9` (b) `buildings_5`

Figure 14: Example KNN results using mean, standard deviation, minimum, maximum and median

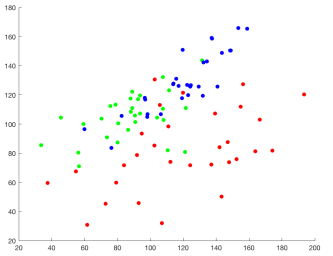
The next step of experimenting involves adding color features to the feature space. The filters of the LM filter bank were all applied on gray scale images and ignore color differences. Thus, we add three new features for each image: the mean values for all color channels red (R), blue (B), and green (G). It turns out that these 3 features are well suited to separate the classes, as can be seen in figure 15.

Figure 16 shows the results of the KNN algorithm when only using the three mean color values as features. The results are very good for the forest example, but much less so for the buildings example that now gets confused with a variety of forests, buildings and sunsets.

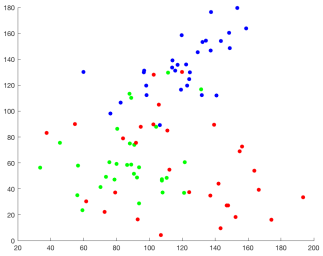
When adding the means of responses back to the filter set, the results for the forest decrease, but the retrieved images for `building_5` are significantly better, as can be seen in figure 17.

The best results for the two examined cases are obtained when using a combination of standard deviations of responses to the texture filters and the color features, as depicted in figure 18

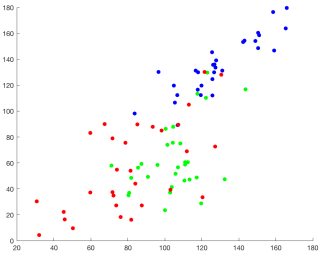
In addition to looking at individual examples of knn-search results using different sets of features, we tried to quantify the quality of a feature set by measuring, how often the 9 images retrieved for each image belong to the same class as the queried image, and how often they come from other classes. Based on that we can calculate the accuracy for different



(a) R channel vs G channel

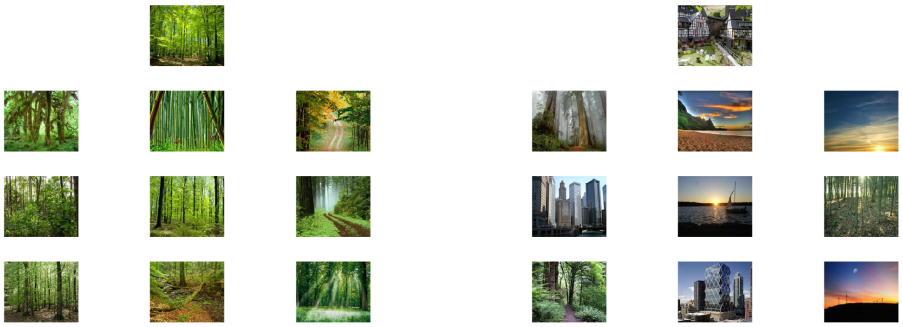


(b) R channel vs B channel



(c) B channel vs G channel

Figure 15: Separability of RGB features



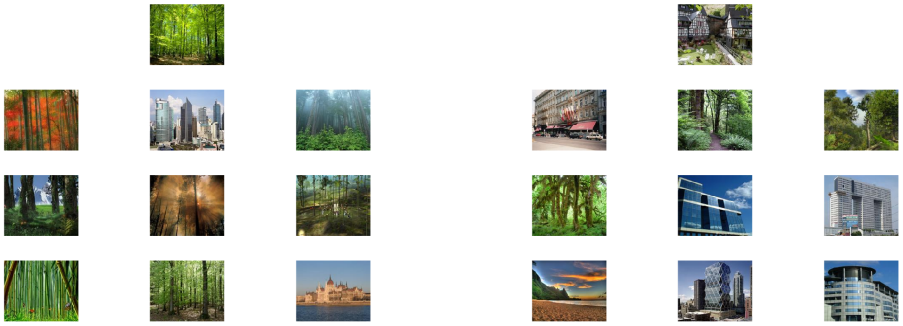
(a) `forest_9`

(b) `buildings_5`

Figure 16: Example KNN results for RGB based features

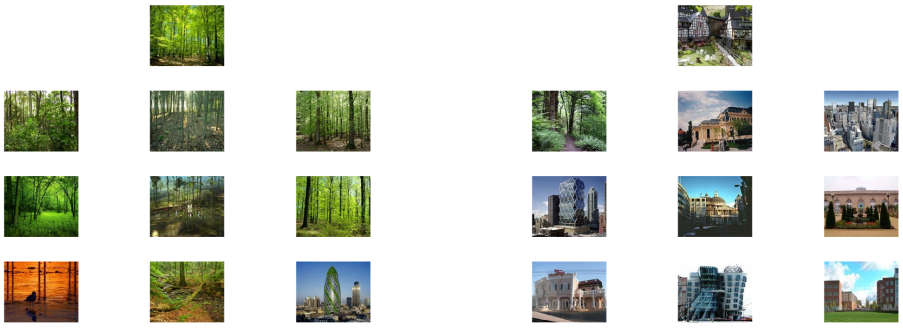
choices of features. It is important to note that this metric does not actually provide the accuracy of retrieving similar images, but only the accuracy of retrieving images of the same class. One can argue, however, that these metrics would yield at least similar results. In the following table we summarize the results of this experiment:

As we can see in table 1, the accuracy of classification varies quite a lot based on which features we select to be included in the knn-search. If we only choose one aggregation method for the filter responses, standard deviation achieves the highest accuracy with 63% of retrieved similar images actually belonging to the same class as the queried image. One surprising observation is, that simply using the mean value of each image's color channels RGB yields by far the highest accuracy of 76%. Adding any kind of aggregated texture feature decreases the results for this metric. It is likely that a learned weighting of color and texture features would result in even higher accuracy, but this was sadly beyond the scope of this report. If a single texture feature had to be selected, the obvious choice for the



(a) `forest_9` (b) `buildings_5`

Figure 17: Example KNN results for mean and RGB based features



(a) `forest_9` (b) `buildings_5`

Figure 18: Example KNN results for standard deviation and RGB based features

mean	stdev	min	max	median	RGB	accuracy
yes	no	no	no	no	no	0.42
no	**yes**	no	no	no	no	0.63
no	no	**yes**	no	no	no	0.58
no	no	no	**yes**	no	no	0.53
no	no	no	no	**yes**	no	0.40
no	no	no	no	no	**yes**	0.76
yes	no	no	no	no	**yes**	0.53
no	**yes**	no	no	no	**yes**	0.71
no	no	**yes**	no	no	**yes**	0.59
no	no	no	**yes**	no	**yes**	0.69
no	no	no	no	**yes**	**yes**	0.50
yes	**yes**	**yes**	**yes**	**yes**	no	0.46
yes	**yes**	**yes**	**yes**	**yes**	**yes**	0.51

Table 1: Comparing classification accuracy for different feature sets

given three classes would be the standard deviation - it achieved the highest results both with RGB features and without them.

Appendices