Content-based retrieval of visual information
Oerlemans, A.A.J.

Chapter 8

Texture Classification: What Can Be Done with 1 or 2 Features?

For real-time imaging, pattern classification with short feature-vectors is desirable. One well-known and effective method used for texture classification is the use of statistical information on 3x3 pixel blocks. Our method is an extension of this method to larger and non-square features, which we call 'constructed features'. The standard 3x3 features are very usable for real-time imaging, because the resulting feature space can only contain 512 elements and the vectors can be created very quickly. However, larger or non-square features might give better classification results. Our proposed method searches for small sets of constructed features, with arbitrary size and shape, which will give the best results for a classifying a specific texture, and still keeping the feature vectors as short as possible. In this chapter, we show the selected features and the performance of our method with a minimum distance classifier and with a neural network on a texture classification task.

8.1   Introduction

In many applications such as video analysis, computational efficiency is of critical importance. Thus, in certain contexts, the maximum allowed computational complexity is known beforehand and the algorithm must therefore be designed to accommodate the computational requirements.

Furthermore, the novel aspect of this chapter is in exploring binary pattern oriented features which are not simply 3x3 templates. We attempt to find more

general template shapes which could give better performance and accuracy than the traditional 3x3 features.

Our texture classification method generates features for a specific texture and iteratively selects the best features to use. In this way, a feature set of any size can be created, that will give the best classification results for that specific texture.

8.2 Related work

In 1990, Wang and He have introduced the Texture Unit (TU) [93], which is generally considered as the basis for texture analysis based on the distribution of 3x3 pixel blocks.

A variant of the TU was introduced by Ojala et al. [60], which they named the local binary pattern (LBP). They showed that classifying textures using LBP features turned out to be very effective, especially when combined with a local contrast measure. Later they have improved the LBP feature by creating a rotation invariant version of it [62].

Other texture unit variants include multiscale selected local binary features by Raja [64] or the simplified texture unit by Madrid-Cuevas [45]. Variants of local image features have also been applied to other fields, for example face detection [33] or facial expression recognition [26].

In 1996, Huijsmans et al. [30] showed that using frequencies of 3x3 binary patterns is very helpful in copy-locating in image databases. In their experiments, they showed that using a subset of all features or non-equal weighing yielded better results than using the entire set of features with equal weights. Mäenpää later found the same results for the LBP features [46].

In line with these findings, we expect that non-square features or larger features might also improve better classification results over the use of the entire range of 3x3 features.

8.3 Our method

We define a "Constructed" feature as a binary pattern of NxN in size. An example is shown in Figure 8.1. All these constructed features are tested for classification accuracy.

In general, the method applies to arbitrarily large features. For computational efficiency reasons, we note that the space of all potential features is exponential, which is not feasible for straightforward searching. To make the problem tractable, we record all possible features within the training set up to a certain size and then we select features which appear in all images of the training set. These features form the space for candidate features.
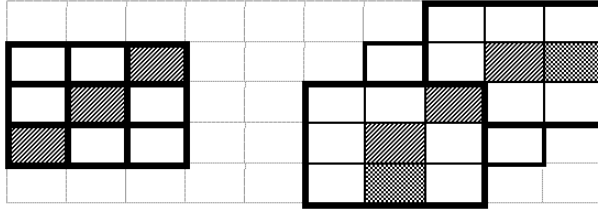
Figure 8.1: Constructing a feature. Starting with a 3x3 feature in a), we can then create a new feature in by joining several 3x3 features together to 'construct' a 5x5 feature as in b).

For our first experiments, we used an even more restricted set of features: each of the features in the candidate set for a specific class was required to be present in every image from that class. We wanted to make sure we were using the most descriptive features, which are probably those features that can be found in every positive example.

To get binary images for our feature construction, we create edge-detected versions of our textures with the Marr-Hildreth algorithm. This method detects intensity changes in an image by convolving it with the $\nabla^2 G$ filter, where ∇^2 is the Laplacian operator

$$\left(\frac{\partial^2 x}{\partial x^2} + \frac{\partial^2 y}{\partial y^2} \right) \tag{8.1}$$

and G stands for the two dimensional Gaussian distribution

$$G(x,y) = e^{-\frac{x^2+y^2}{2\pi\sigma^2}} \tag{8.2}$$

In effect, this is a 'mexican-hat' operator. The zero-crossings in the output of this convolution mark the intensity changes. Pixels with an intensity change, or zero-crossing, are set to 1 in the resulting image and the rest of the pixels are set to 0. Figure 8.2 shows an example of this edge detection method.

The sigma parameter that is used for the Marr-Hildreth algorithm selects the scale the method works on, or the size of the mexican hat. Higher values yield the detection of less detailed structures in the images. In our experiments we used a fixed value of 2.0 for sigma, but by varying this parameter, the search space for the best features could even be extended.

For each feature, we keep track of the number of times it is found within each image. Note that the number of features that can be found for each class is significantly less than the maximum number of possible features. For computational purposes, the features are stored in hash tables for fast lookups.
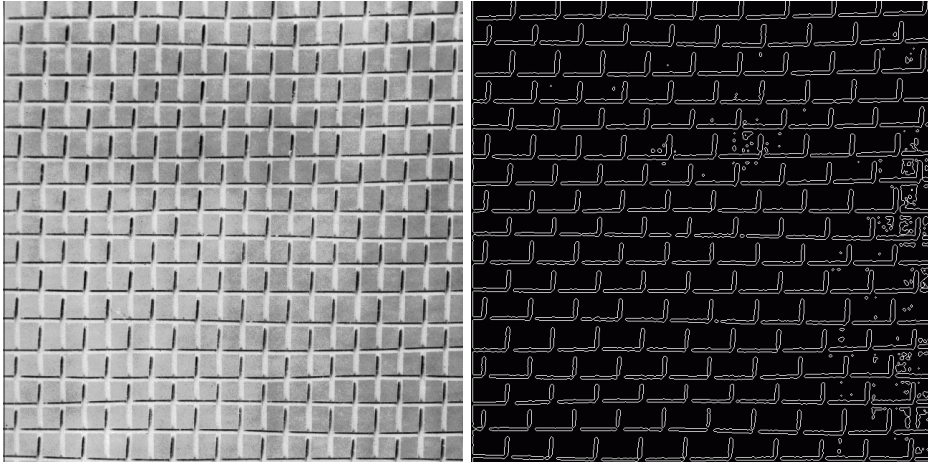
Figure 8.2: Brodatz texture D1 and the result of the Marr-Hildreth edge detection applied to it with sigma = 2.0

Since we are trying to find an optimal feature set, in theory every possible combination of features should be tested for classification accuracy, but the required processing time will increase exponentially. We therefore use a greedy hill-climbing method by iteratively increasing the feature set by adding a feature to the set that minimizes the training error.

The first feature can be selected directly by using the feature that minimizes the training error. After that, in case that x features are already found for a specific class, we test the classification accuracy of all sets of features that consist of these x features plus one of each of the features that are left. By repeatedly following this procedure, a best feature set of any length can be found.

8.4   Results

For testing our method, we used 12 textures from the Brodatz database, which are listed in Figure 8.3. We created a separate classifier for each texture class.

For each of these classes, we have generated random subblocks of size 64x64 and we selected 50 training blocks from the upper half of the image and 50 test blocks from the lower half. For this experiment, we have limited the size of the constructed features to 7x7.

Table 8.1 contains the results from our first experiments. We have determined the best two features for each class, using 3x3 features only and with the constructed features. Also, we show the comparison between minimum distance classifiers and neural network classifiers. The neural network is a fully connected 3-layer

network that has a hidden layer with 10 nodes and one output node. These values were determined by empirical testing. The number of input nodes is equal to the number of features.

Note that for each class, there are 50 positive examples of that class and 550 negative examples in both the training and the test set. The misdetection rates are given over all test images.

Class	Nearest neighbor (1st and 2nd feature)				Neural network (1st and 2nd feature)			
	3x3		constructed		3x3		constructed	
D1	0.043	0.002	0.008	0.002	0.035	0.005	0.083	0.005
D3	0.095	0.103	0.070	0.077	0.087	0.102	0.065	0.068
D4	0.072	0.047	0.072	0.047	0.052	0.040	0.063	0.042
D6	0.018	0.023	0.005	0.027	0.027	0.028	0.003	0.032
D9	0.180	0.107	0.180	0.107	0.047	0.060	0.047	0.060
D11	0.097	0.108	0.080	0.073	0.100	0.098	0.078	0.055
D16	0.057	0.102	0.057	0.047	0.050	0.053	0.042	0.042
D17	0.060	0.058	0.055	0.058	0.052	0.085	0.047	0.053

Class	Nearest neighbor (1st and 2nd feature)				Neural network (1st and 2nd feature)			
	3x3		constructed		3x3		constructed	
D20	0.060	0.008	0.000	0.017	0.060	0.007	0.083	0.007
D21	0.000	0.000	0.000	0.000	0.003	0.000	0.000	0.020
D24	0.088	0.097	0.088	0.097	0.080	0.012	0.083	0.012
D29	0.083	0.115	0.097	0.088	0.102	0.090	0.102	0.065

Table 8.1: Misdetection rates and the selected features for 3x3 features only and for constructed features, using a minimum distance classifier or a neural network.

8.5   Discussion, conclusions and future work

The first thing that is noticeable in the results, is that adding a second feature does not always give a better test result. This is probably caused by the small number of images in the training and test sets. In general, the training error was lower with two features.

Also, it is interesting to see that for some textures, the optimal features are still the 3x3 features, even when all constructed features are considered. The reason for this could be the fine scale of the texture. The small details might have are already been captured by the 3x3 features.

For the minimum distance classifiers with one feature, 8 out of 12 selected features were constructed features. For the neural network classifiers with one feature, 9 out of 12 selected features were constructed features. These features were selected based on a lower misdetection rate on the training set.

Table 8.2 shows a comparison between constructed features and 3x3 features. It can be seen that using the constructed features gives an improved classification result on average.

	MDC-1	MDC-2	NN-1	NN-2
Underperforming	1	2	4	3
Equal	5	6	2	4
Outperforming	6	4	6	5

Table 8.2: The performance of the constructed features compared to the 3x3 features.

For us, this justifies the idea of further experimenting with constructed features instead of 3x3 features for texture classification.

Our next experiments will focus on the constructed features with what we call d̈on't carep̈ixels. After constructing a feature, we will test the same feature with one or more pixels left out. We expect these features to outperform the features shown in this chapter.
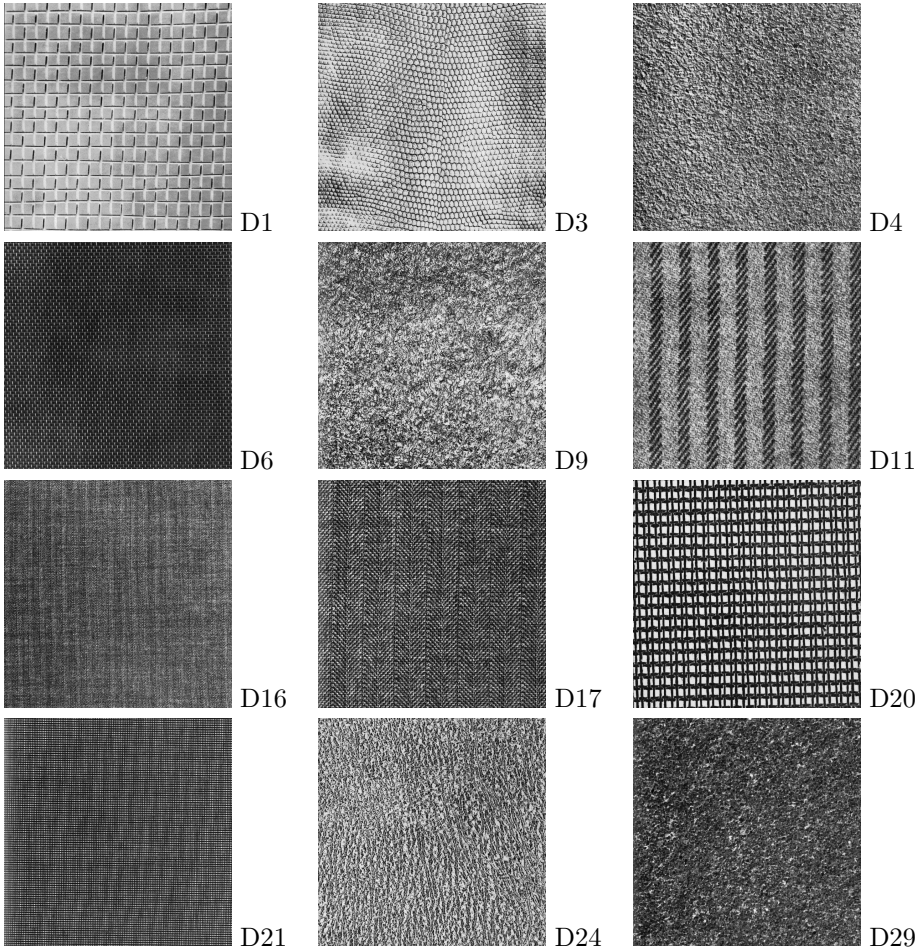
Figure 8.3: Brodatz textures used in our experiments.