Laboratory 5: features

Ramón Emiliani
Universidad de los Andes
201125694
rd.emiliani689@uniandes.edu.co

Alejandro Posada
Universidad de los Andes
201227104
a.posada10@uniandes.edu.co

Abstract

Texton dictionaries are an useful representation that allows to classify images according to their texture. We used the Ponce group's texture dataset to build a texton dictionary and the corresponding texton representations of the images. We used two classifiers (K-nearest neighbors and random forests) to classify the test images according to their texton representation. We found that the random forests (accuracy = 44.8 %) outperformed the K-NN classifier (accuracy = 41.2%).

1. Introduction

Image textures are collections of patterns that tend to show repetition. Texture is an important cue to object identity and material properties [1]. This explains why many of modern object recognition algorithms are built around texture representation [9]. The main kinds of texture representations are local representations (encoding the texture very close to a point in an image), pooled representations (describing the texture within an image domain) and data-driven representations (modeling a texture from a region of an example) [1].

Pooled texture representations deal with the challenging problem of texture clasification, which aims to determine what texture is represented by a patch in an image. Popular approaches to this problem is analyzing statistics of the responses of filter banks [8][3]. In particular, textons have been widely used in the last decades to perform texture clasification [6][7][5]. Textons are representations of small texture patches (by, for example, a collection of filter bank responses) and can be viewed as the basic constituant element of textures.

Texton-based texture classifiers use texton dictionaries to classify textures. These dictionaries are composed of a set of clusters built out of a training set of vectors that represent the filter responses. Many different clusterers can be used for vector quantization (i.e. representing vectors in a continuous space with numbers from a set of fixed size [1]), the most popular being k-means and its variants. Then, texton-bases classifiers use texton histograms that measure the frequency by which textons from the dictionary appear in the texture [9]. Finally, texture classification is performed by means of classifiers such as nearest neighbors classifiers, support vector machines or random forests that are trained on the texton frequency histograms [9]. In this laboratory, we used the images from the Ponce group's texture database, represented them as textons and trained and evaluated two classifiers based on the texton representation.

2. Materials and methods

2.1. Database

We used the Ponce group's texture database [4]. It features 25 texture classes with 40 samples each. It is divided into 30 training images per class (750 images) and 10 test images per class (250 images). We divided the default test images into validation and test sets (125 images each). The images are in grayscale JPG format, 640×480 pixels. Figure 1 shows an example of the database's images.

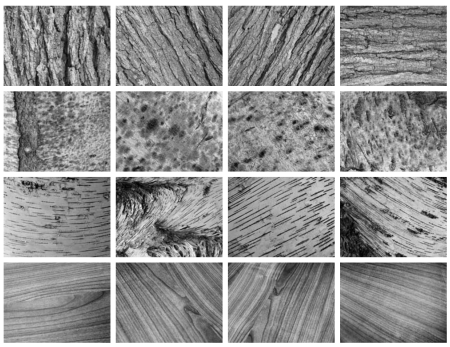


Figure 1: Example texture images from the database.

2.2. Texton dictionary

In order to generate the filter responses, training images were convolved with a filter bank. We selected 3 training images per class, giving a total of $25 \times 3 = 75$ training im-

ages. These were chosen in such a way that they sampled the variations in viewpoint and illumination. These images were horizontally concatenated to create a large image. Two texton dictionaries were created by convolving this image with a filter bank and clustering the filter responses via k-means with $k = 50$ and $k = 125$. However, both classification algorithms performed worse with the $k = 50$ than with the $k = 125$ dictionary. Thus, we decided to use the dictionary with 125 textons. The criteria to choose k was that it should be a multiple of 25 (number of classes) so that, ideally, each class would be represented by the same number of textons — in this case, 2 or 5 (this is, of course, unlikely, but this method encourages each class to have ta similar number of textons). The learnt textons were collected into a single dictionary. The filter bank was created with the Matlab function `fbCreate`, implemented by David R. Martin (as well as the other functions used to generate the dictionary), using the default parameters (8 orientations, $\sigma_{start} = 1$, 2 scales, scaling $= \sqrt{2}$ and elongation $= 2$). Figure 2 shows the obtained filter bank.

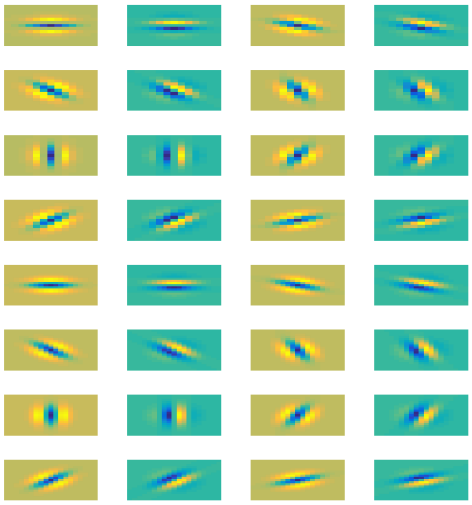


Figure 2: Filter bank used to create the texton dictionary. It is composed by 13×13 and 19×19 filters.

Given that some textures are defined by large thick lines and other textures by thin an small lines, a small filter can produce responses in both kinds of textures, while a bigger filter doesn't respond to the "small textures". A filter is discriminatory if, when applied in images of different textures, produces a greater variance in the response. Thus, it is very likely that the 19×19 filters are more discriminatory than the 13×13 ones.

2.3. Models

After learning the texton dictionary, we generated the models corresponding to the training images. This was done by convolving them with the filter bank an then labelling each filter response with the texton that lies closest to it in filter response space (the function that performs this step is `assignTextons`, provided also by the Berkeley Vision group). The frequency with which each texton occurs in the labelling (i.e. the histogram of textons) forms the model that corresponds to the training image.

2.4. Classification

In the classification stage, the same procedure was followed to build the histograms corresponding to the novel images. These novel histograms were then compared to the histograms learnt during training. We tried two different classifiers: K-nearest neighbors and random forests.

2.4.1 K-nearest neighbors

KNN algorithm evaluates the available cases and classifies new cases according to a similarity measure, such as distance functions. A new case is classified depending on the votes of its neighbors, i.e. the case is assigned to the most common class amongst its K nearest neighbors. The chosen distance function used to measure the similarity between the novel and learnt histograms is the chi-squared distance:

$$d(v, w) = \sum_{i=1}^{N} \frac{v_i - w_i}{v_i + w_i} \tag{1}$$

where v and w are the histograms, v_i and w_i are the bins and N is the number of bins.

In order to find the optimum number of neighbors (K), we tested odd values for K between 1 and 49 in the validation set.

2.4.2 Random forests

A random forest is a collection of simple decision trees, i.e. binary trees where every non-terminal node tests one or more attributes and the result is used to select the branch to follow from that node. Random forests use bagging (or bootstrap aggregation): given a dataset S with N points, a *bootstrap sample* S^* is defined as the random selection with replacement of N points from S (the same point might appear several times in $S*$ and some points present in S might not appear in $S*$) [1]. Bagging is the process of building B bootstrap samples, then growing a decision tree for each one and classifying by using a majority vote among the decision trees. Additionally, random forests randomly select a subset of the input variables at each recursive step during

the training step (feature bagging) [1]. Thus, the correlation between the decision trees is reduced.

The parameters we considered when building the random forests are the number of trees and the minimum number of observations per tree leaf. The values we used to test the number of trees are 1, 10, 20, 30 ... 150 and we tested with 1, 2 and 3 as the minimum number of observations per leaf.

2.4.3 Evaluation

We evaluated the performance of our algorithm on the test set (125 images) with the optimum K for K-NN and the best values of random forest's parameters. The evaluation metric used is the accuracy, that corresponds to the average of the diagonal of the normalized confusion matrix.

3. Results

We found that the optimum number of neighbors to be used in the K-NN algorithm is 1. As figure 3 shows, the highest accuracy corresponds to K $= 1$. This result is explained because the model uses only the training point closest to the query point, so the bias of is low but the variance is high [2]. Thus, the model is prone to overfitting and a different sampling of the training set might cause a considerable divergence from the results found with the actual training set.

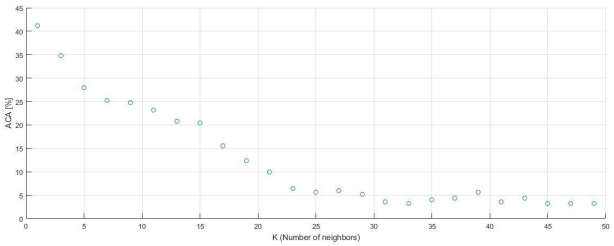


Figure 3: Accuracy with different values of K for the K-NN algorithm.

Figure 4 shows the confusion matrix of the K-NN algorithm on the test set with K$= 1$. We obtained an accuracy of 41.2%.

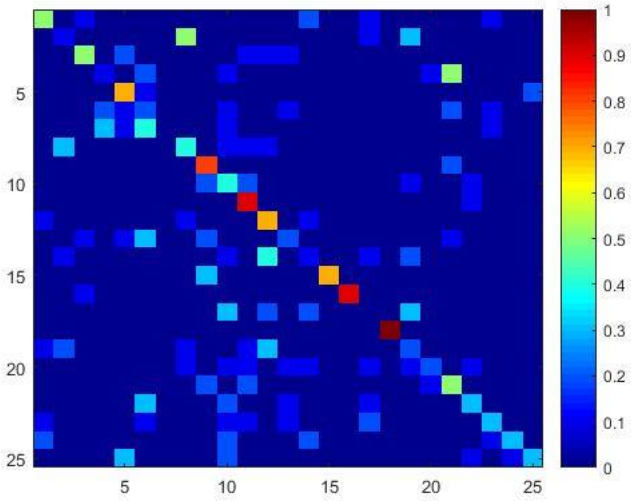


Figure 4: Confusion matrix obtained on the test set using K-NN with K$= 1$ (accuracy $= 41.2\%$).

The experiments on the validation set with the random forests show that as the number of trees grow, the accuracy grows too. The highest accuracy (50 %) was obtained with 150 trees and a minimum of 2 observations per leaf (figure 5).

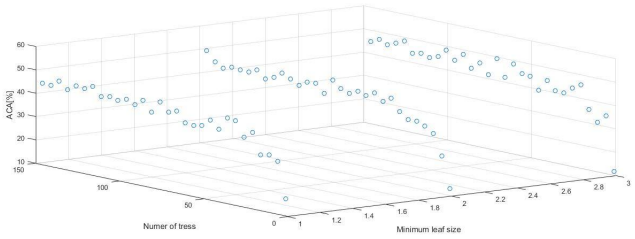


Figure 5: Accuracy with different values of trees and minimum number of observations per leaf.

Figure 6 show the confusion matrix of the results of the random forests algorithm on the test set using 150 trees and a minimum of 2 observations per leaf. With this method we obtained an accuracy of 44.8 %.

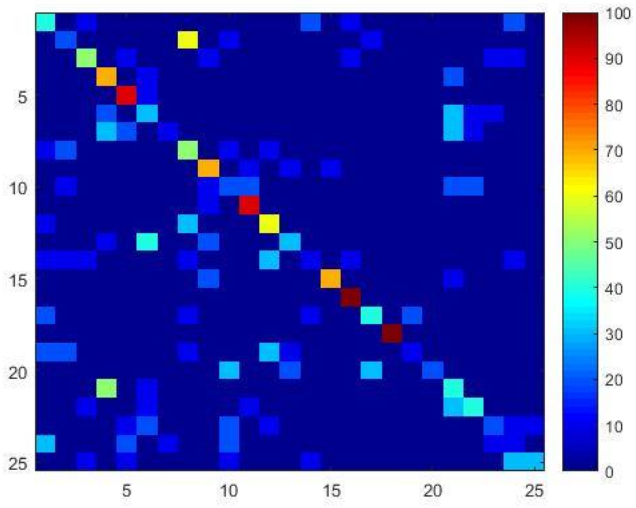


Figure 6: Confusion matrix obtained on the test set using random forests with 150 trees and a minimum of 2 observations per leaf (accuracy $= 44.8\%$).

Table 1 shows how much time it took to create the texton dictionary ($k = 125$) and to train and apply each classifier. The construction of the texton dictionary is the longest process and the longest sub-process corresponds, by far, to the texton calculation (almost 2.5 hours).

Table 1: Times taken by each process

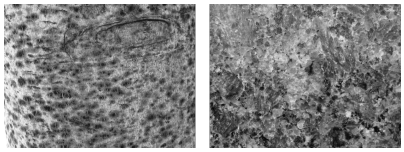
Process		**Time (s)**
Texton dictionary	Concatenate images	4.21
	Calculate textons	8868.91
	Assign textons	67.50
K-NN		165.21
Random forests		231.95

Comparing the results of both classifiers, we see that the random forests algorithm (accuracy =44.8%) has a better performance than the K-NN algorithm (accuracy =41.2%). However, K-NN is 28.77% faster than random forests (table 1). Thus, given that the difference between both accuracy is of 3.6%, if time is considered as an important factor, K-NN is a good alternative to random forests.

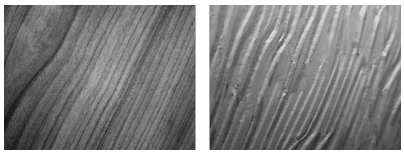
Figure 4 shows that the categories that are more easily confounded by the K-NN classifier are: 4 (wood 1) and 21 (wallpaper), 13 (wall) and 6 (wood 3), 2 (bark 2) and 8 (granite), 14 (brick 1) and 12 (pebbles), and 9 (marble) and 15 (brick 2).

Figure 6 shows that the categories that are more easily confounded by the random forests method are: 21 (wallpaper) and 4 (wood 1), 13 (wall) and 6 (wood 3), 2 (bark 2) and 8 (granite), 6 (wood 3) and 21 (wallpaper), 7 (water) and 21 (wallpaper), and 22 (fur) and 21(wallpaper).

In both cases, there are two pairs of classes that are confounded by the algorithms 50% or more of the times: classes 2 (bark 2) and 8 (granite), and 4 (wood 1) and 21 (wallpaper) (figure 7). Both algorithms confound these textures because they are described by similar texton histograms. This is due to the fact that these textures present structures such as blobs and dots (figure 7a) and oriented lines (figure 7b) that have similar responses to the filter bank.



(a) Bark and granite.



(b) Wood and wallpaper.

Figure 7: Texture classes confounded 50% or more of the times by the random forests classifier.

4. Conclusions

We built and used a texton dictionary together with two classifiers in order to categorize images according to their texture. One of the limitations of our method is the number of parameters we chose to vary in the random forests algorithm. Given that this method has many different parameters, it is possible to tune and upgrade the results by altering other parameters such as the cost, the number of variables to select at random for each decision split, etc. This limitation does not apply to the K-NN method since it has few parameters; however, this algorithm could not outperform the random forests and is very sensible to similarities between classes. Additionally, other types of classifiers, such as Support Vector Machines, could be more succesfull at classifying the textures.

The quality of the texton dictionary is limited by the number of training images. Thus, using a larger set of images would result in a more accurate texton description of the images. Taking into account that the slowest process in our method is calculating the textons, a substantial improvement could be using a faster clustering method such as fast k-means, DBSCAN or OPTICS. Other improvements to the method could be varying the filter bank (filter sizes and including circular filters) and comparing the results with other standard filter banks. Also, it is very likely that preprocessing the images (increasing the contrast, for example) could result in a better differentiation between texture

classes.

References

[1] D. Forsyth and J. Ponce. *Computer vision*. Pearson, 2nd edition, 2015.

[2] T. J. Hastie, R. J. Tibshirani, and J. H. Friedman. *The elements of statistical learning : data mining, inference, and prediction*. Springer series in statistics. Springer, New York, 2009. Autres impressions : 2011 (corr.), 2013 (7e corr.).

[3] A. K. Jain and F. Farrokhnia. Unsupervised texture segmentation using gabor filters. *Pattern recognition*, 24(12):1167–1186, 1991.

[4] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, 2005.

[5] T. Leung and J. Malik. Recognizing surfaces using three-dimensional textons. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1010–1017. IEEE, 1999.

[6] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International journal of computer vision*, 43(1):29–44, 2001.

[7] J. Malik, S. Belongie, J. Shi, and T. Leung. Textons, contours and regions: Cue integration in image segmentation. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 918–925. IEEE, 1999.

[8] J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision*, 40(1):49–70, 2000.

[9] L. van der Maaten and E. Postma. Texton-based texture classification. In *Proceedings of the Belgium-Netherlands Artifical Intelligence Conference*, 2007.