Textons

Johana M. Ramirez Borda
Los Andes University
jm.ramirez11@uniandes.edu.co

Francisco A. Rozo Forero
Los Andes University
fa.rozo1843@uniandes.edu.co

Abstract

A texton is the representation of a small texture patch, for example a collection of filter bank responses, that is usually expressed in terms of a frequency histogram. These histograms are used as trainning data in classifiers and then the test images are classified using them. So, the present project used the ponce group textures database in order to represent those images using textons and then train and evaluate two classifiers (K nearest neighbour -KNN- and random forest -RF) based on the texton representation. It was found out that the best combination of parameters was number of textons 175*, number of trees* 75*, distance metric standardized Euclidean and a proportion of* 70% *train and* 30% *test images. When using these parameters, the performance of both classifiers was measured with the confusion matrix and the ACA index. KNN method resulted in an ACA of* 0.73 *and RF method resulted in an ACA of* 0.83*, being this one the best.*

1. Introduction

Texture analysis is a common task in computer vision and refers to the characterization of regions in an image by their texture content. In this sense, the texture is related to roughness, bumpiness, smoothness..etc which quantitatively refers to variations in the gray levels of an image. This analysis is used in numerous applications such as medical image processing, automated inspection, remote sensing and texture segmentation. Even more, texture analysis is used when trying to identify objects in an image that are more characterized by their texture than by intensity [6].

A very frequent approach to texture analysis is based on the study of the responses of the images to a filter bank. So, within this context, the concept of *textons* emerged. A texton is the representation of a small texture patch, for example a collection of filter bank responses, that is usually expressed in terms of a frequency histogram. This histogram measures the relative frequency by which textons from the codebook apper in the texture [9]. So, once aqcuired the frequency histogram of textons, texture classification is performed by classifiers such as nearest neighbour (NN), suport vector machines and random forest (RF).

Considering this, the objective of the present project is to represent images using textons and then train and evaluate two classifiers (KNN and RF) based on the texton representation.

2. Materials and methods

For this lab, the software implemented was MATLAB R2016b.

2.1. Database

The database used in this project was provided by the Ponce group and was downloaded from [5]. The creation of this database was supported by the National Science Foundation, the European project LAVA, the UIUC-CNRS Research Collaboration Agreement, the UIUC Campus Research Board, and the Beckman Institute.

This is a texture database that contains 25 texture classes, 40 samples each. All images are in grayscale, .jpg format and 640x480 pixels. These classes are presented in Figure 1.

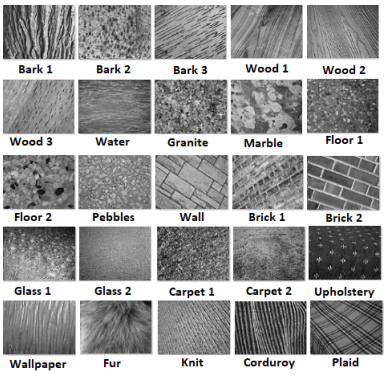


Figure 1. Example of one random image within each category of the database.

As it can be seen, there are categories such as bark, wood, water, floor, carpets,glass,wallpaper, etc .Also, each

category within the database has images taken from different angles and distances.

2.2. Image Pre-processing

Before the texton processing and assignation, a random set of images from the database was chosen to serve as the base to make the texton library. From each category, were picked at random two images so they could represent their set. However, as the textons requires an unique image that contains all the concatenated images, using the entire image would had made an image too big for processing (1280x12000). Then, a small portion of each of the chosen images was concatenated. A window of 40x40 from the centre of each of the images was used, because most of them contain more characteristics from the images than in the border.

Moreover, three different proportions of train/test images were tested: 50/50, 30/70 and 70/30, in order to prove whether the proportion of trainning images over test images affects the performace of the classifier.

2.3. Textons processing

The texton representation of an image allows to identify and quantify it by its textures. In addition, this representation is useful when trying to classify images that are more characterized by texture than intensity. To do so, a textons library is created. Then, this library is used to acquire the frequency histogram which gives information about how many times a texton is present in an image. This histogram is the called "feature" of the image and is used to train the classifier. So, in both cases (KNN and RF), the histograms of the training images are provided as well as their annotations. Then, when trying to classify a test image, the classifier will compare its histogram to those saved and will assign a category to the test image.

Therefore, the process used to obtain the dictionary is shown in Figure 2.

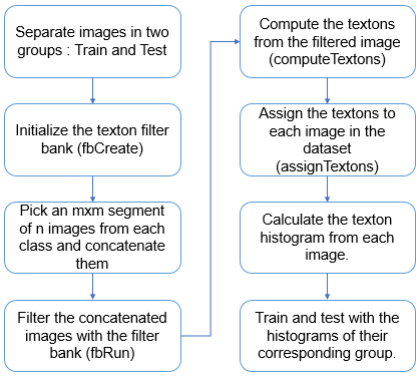


Figure 2. Algorithm used to create the dictionary.

When creating the textons library, several numbers of textons were tested : 5,10,25,50,75,100,125,150,175,200 and 250. We decided to test more than one value in order to prove whether the number of textons affects the performace of the classifier. In the *Results* section it will be shown which one of these values of number of textons improved the classification.

On the other hand, as explained before, in order to study texture, it is usually necessary to consider the response of the images to a filter bank. In this sense, Figure 3. shows some examples of the filters found in the bank obtained when using the function *fb_Create* that was provided.

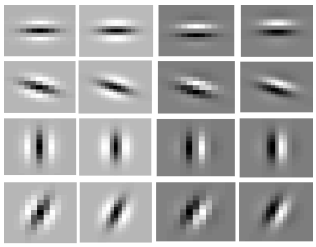


Figure 3. Examples of filters used in the filter bank to study texture.

Analyzing Figure 3. it is valid to say that even though there is more than one filter for the same category (horizontal, vertical, right diagonal, left diagonal), they differ one from another regarding the amount of contrast within each. For example, the intensity changes in some of these filters are greater than in others: so, those images filtered with filters of more contrast may detect less number of textures but it will do it more accurately. On the contrary, filters with less contrast will detect much more textures but some of them could be noise or intensity changes that dont necessarily belong to a texture pattern. Therefore, the first type of filter will be more discriminative than the second.

Moreover, one filter will be more discriminative than other depending on the texture of the image that is being filtered with it. For example, a filter that enhances horizontal borders used between images that contain the same (or similar) amount of these type of borders will produce a very similar response map for both images. Then, they wont be successfully classified. This situation will also occur when testing a vertical-border filter, diagonal, etc. For this reason, in order to obtain response maps that truly discriminate of differentiate images from others, it is necessary to use a bank of filters (as the one used) that will enhance the complete group of characteristics of each image because it detects more than one type of texture.

2.4. KNN classifier

K nearest neighbour algorithm finds a centroid for n categories based on a set of points; it clusters the information

based on their distance. For the training, a random set of images from the database was filtered through textons and then the corresponding textons histograms were used as the data used for training.

Besides selecting the training images at random, and varying the proportion of images, the distance measurement method was modified to see which one fit the most to this dataset using the knn algorithm. The results obtained when using each one are shown in Table 4.

2.5. Random Forest Classifier

Random forest is an algorithm that by taking different characteristics of an input, it classifies it by using probabilities. The algorithm consists on taking different subsets of images making trees out of them, whose nodes are a characteristic and the arcs the probability of having it. Then, for the evaluation, an input passes through an initial node, then branching unitl it reaches a category. Finally, depending on where the input landed on each tree, it's category is chosen.

After trainning and evaluating both classifiers using the randomly selected test images, the function *confusionmat* and the ground truth data were used in order to visualize the performance of each one and then, the ACA index was calculated by normalizing (by columns) this matrix and calculating the mean of the resulting diagonal. This index is relevant since the diagonal of the confusion matrix gives information about the number of correctly classified images and the mean gives information about the overall classification accuracy.

With all of this in mind, the steps used in the methodology of this laboratory are the following: First,

1. Select the number of textons that improves the performance of both classifiers by varying this parameter, then calculating the ACA index in each case and comparing them.
2. Select the number of trees that improves the performance of the random forest (RF) classifier by varying this parameter, using the parameter found in the previous step, then calculating the ACA index and comparing them.
3. Select the distance metric that improves the performance of the k-nearest neighbour (KNN) classifier by varying this parameter, using the parameters found in the previous step, then calculating the ACA index and comparing them.
4. Select the proportion or ratio between $train/test$ images that improves the classification by varying between $50\%/50\%$ (used as control in the previous steps), $30\%/70\%$ and $70\%/30\%$. In this steps the values of number of textons and threes found before were used. The ACA index was also used to compare.
5. Having found all the parameters that improved the performance of both classifiers, all of them were used in order to test whether KNN or RF produced the best results. This means that the confusion matrix and the ACA index were used to find the best classifier.

3. Results and discussion

3.1. Selection of number of textons

Firstly, the number of textons used to create the textons dictionary was varied between 5, 10, 25, 50, 75, 100, 125, 150, 175, 200 and 250. The parameter *distance* in the method KNN wasn't modified so the one used was *euclidean*, which is the default in the matlab function. Finally, the parameter *number of trees* was set in 50 as an intermediate value because in the next subsection it will be varied between a range of numbers smaller and larger than 50.

Table 1. Variation in number of textons. The number of trees in RF was set in 50 and the distance in the KNN method was Euclidean (default).

Number of texons	ACA test KNN	ACA test RF
5	0,31	0,37
10	0,33	0,39
25	0,51	0,61
50	0,60	0,68
75	0,65	0,74
100	0,70	0,77
125	0,68	0,80
150	0,64	0,75
175	**0,71**	**0,812**
200	0,69	0,805

As it can be seen in Table 1., the value of number of textons that improved the performance of both classifiers was 175, resulting in an ACA of 0,71 for KNN and 0,81 for RF. For that reason, the number of textons selected was 175.

Then, while varying the number of textons, the time that it takes to create the texton dictionary was measured. The results are shown in Table 2.

Table 2. Time that it takes to create the texton dictionary depending on the number of textons used.

Number of texons	Time of creation of texton dictionary [min]
5	20,7
10	20,8
25	35,1
50	38,2
75	40,0
100	44,6
125	44,8
150	54,1
175	55,3
200	54,0
250	57,4

In Table 2. it can be seen that the time that it takes to create the texton dictionary is directly proportional to the number of textons used. This is predictable since the more words or textons wanted to be included in a dictionary, the more time will it take to create that dictionary. However, it is also observed that the minimum time spent was 20 minutes (with 5 textons) and the maximum was 57 minutes (with 250 textons), being both of them time consuming. This may be due to the fact that during the extraction of the textons, each feature has to be compared and assigned to a texton by searching for the nearest texton in the whole library. This process is very time consuming when the library is big [4]. In our case, as we are increasing the size of the library, the time spent is increasing too. Also, the time spent increases because the size of the feature that is being assigned is dimensionally high. In this sense, the process is time consuming not only because of the number of times that the feature has to be compared, but also because such feature is large and each comparison takes a long time. Realizing this issue, one understands that a common exercise used to improve the performance of a classifier, as it is to enlarge the size of the patch and therefore of the feature, will make the algorithm more time consuming [4].

3.2. Selection of number of trees

Second, using 175 textons, the number of trees was varied between 1, 5, 10, 25, 50, 75, 100 and 300. The distance in KNN was still the default. In order to choose the value that improved the performance of the RF classifier, the confusion matrix as well as the ACA index were calculated. Also, the time that it takes to train and apply this classifier was measured. These results are shown in Table 3.

Table 3. Variation in number of trees when 175 textons were used. ACA index and time spent are shown.

Number of trees	ACA train RF	ACA test RF	Time [s]
1	0,80	0,50	1,48
5	0,97	0,61	1,47
10	1	0,71	1,67
25	1	0,79	2,58
50	1	0,81	3,39
75	**1**	**0,83**	3,59
100	1	0,81	4,84
300	1	0,82	10,90

In this table it is possible to see that the performance (ACA index) of the RF classifier increases as the number of trees increases, until it reaches one peak and then starts to decrease. This peak, it means a maximum ACA of 0,83, is reached with 75 trees. Therefore, this value was chosen for this parameter. Currently, even though RF is a widely used method, literature provide few or no guidance about how many trees should be used. In fact, the majority of users set this value based on a trial basis. In our case, the peak and the posterior decrease of performance is similar to some results found in related literature which shows that, at a certain value, the improvement decreases as the number of trees increases. This due to the fact that, there is a point where the benefit gained from learning more trees will be lower than the cost in computation time for learning these additional trees [2].
Moreover, the results agree with the conclusion of related literature that says that at a certain number of trees, only more computational power is spent, for little or no performance gain [7]. This affirmation is exemplified in Table 3, where is shown that even triplicating the number of trees, the ACA index increased in only 0,01 and on the contrary, the time spent was more than doubled.

On the other hand, as explained before, it is understandable that the time spent when training and applying this classifier increases when the number of trees increases. This due to the fact that there are more trees that have to be created and then assessed in order to obtain the proportion or probability of a feature to belong to a certain class.

3.3. Selection of distance metric

Table 4 shows the ACA indexes for KNN classifier when varying the distance metric. The number of textons and number of trees used were the one found before (175 and 75).

Table 4. Variation in the distance metric for the KNN classifier.

Type of distance	ACA train KNN	ACA test KNN	Time [s]
Euclidean	1	0,680	0,861
SEuclidean	**1**	**0,755**	**0,890**
Cityblock	1	0,730	0,864
Chebychev	1	0,614	0,887
Minkowski	1	0,680	0,861

It can be seen that the distance metric that improved the performance of the KNN classifier was *seuclidean* or Standardized Euclidean distance. Therefore, this distance was selected over the others. This results agree with those found in [1], where Chomboon et al. analyzed the performance of this classifier varying 11 distance measures. Between these distances were found Euclidean, mahalanobis, minkoswki, chebychev, cosine, hamming, and standardized Euclidean. They found that the Manhattan, Minkowski, Chebychev, Euclidean, Mahalanobis, and Standardized Euclidean distance measures resulted in similar accuracy results and its performance were greater than that of the other distances.

3.4. Selection of proportion of train/test images

Table 5. Variation in proportion of train/test images.

Textons	Number of trees	Train / Test	ACA test KNN	ACA train RF
175	75	30%/70%	0,60	0,71
175	75	50%/50%	0,63	0,75
175	**75**	**70%/30%**	**0,73**	**0,83**

The results showed that a proportion of 70/30 in the train/test images was the one that improved the ACA index for both classifiers. This result agree with several studies that have found and have used this same proportion in their algorithms. Some examples are Chomboon et al [1] and Gholami et al [3] who used this proportion. The reason behind this division can be based on the fact that more training data will teach the classifier more features to recognize when evaluating, therefore preparing it better, but also, more test data will measure the performance of the classifier with more accuracy in a scenario with a considerably amount of unseen data [8].

3.5. Best classificator

After all this process of selection of those parameters that improved the performance of each algorithm, the selected ones are:

- Number of textons: 175
- Number of trees: 75
- Distance metric: standardized euclidean
- Train/test proportion: 70%30%

Therefore, the results obtained when testing these parameters in both train and test sets of each classifier are shown:

Best combination of parameters in KNN

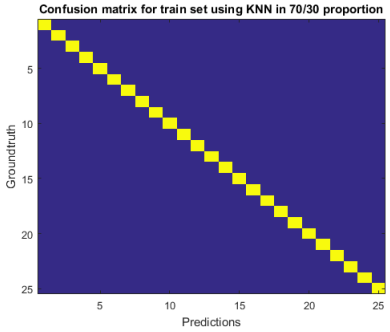
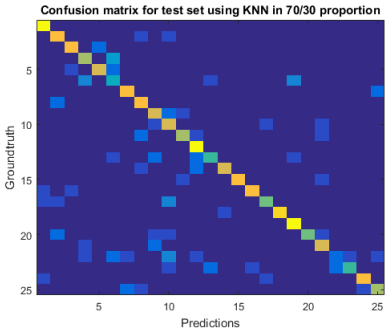


Figure 4. Confusion matrix for the train set using the best combination of parameters in the KNN classifier.

- ACA index: 1

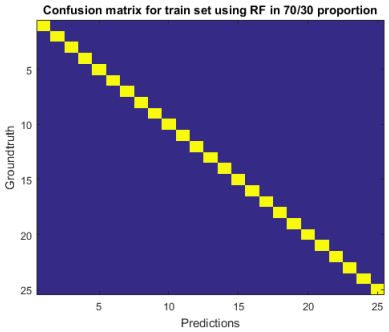


,

Figure 5. Confusion matrix for the test set using the best combination of parameters in the KNN classifier.

- ACA index: 0,73

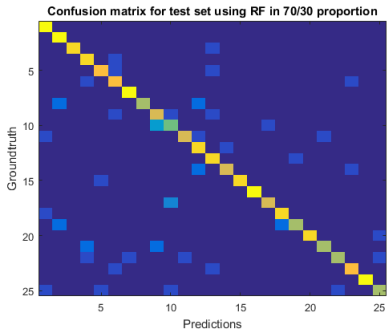
Best combination of parameters in RF



,

Figure 6. Confusion matrix for the train set using the best combination of parameters in the RF classifier.

- ACA index: 1



,

Figure 7. Confusion matrix for the test set using the best combination of parameters in the KNN classifier.

- ACA index: 0,83

The confusion matrices shown above have a colorspace that goes from dark blue to yellow. Dark blue is when there was no classification of an image of a certain class in a category. As the color gets lighter, implies that more images are classified in that category; being yellow the squares that got the maximum images assigned to a category.

With the results presented before, random forest had a better performance than knn. Firstly, it can be seen that with a forest of 25 trees (see Table 3 an ACA of 0.79), the classifier had better performance than the overall knn results regardless of the distance measurement (see Table 4 where the best ACA was 0.76). Moreover, in the confusion matrix of the knn results (see figure 5) is shown that misclassification is more common because the diagonal is less yellow (which means less true positives per category)and light blue spots are more frequent (which means more false positives and false negatives) than the random forest results (see figure 7).

On the other hand, it was observed that the random forest performance increase proportional to the number of trees. However, after a certain number of trees, the performance oscilates without exceeding a maximum value. In consequence, using too much trees increases time and might have similar performances than using an average number of trees (200 according to Feng et al,2015). Each tree is trained with a subset of images and then classifies based on the features of the images. Nevertheless, one tree can't provide a reliable classification, so using a wide set of trees trained with different subsets can provide better results. In contrast, knn doesn't have variability beause it only measures and classifies based on nearness in the space, making it more susceptible to misclassification.

As shown in the confusion matrices in figure 5 and 7, seven pair of classes had similar texton features within them, for both knn and random forest. This means that some images from a pair of classes were classified in those two classes for each pair. In descending order of similarity there were: wood1-wood3, marble-floor1, pebbles-bricks, bark2-granite, floor1-wall, wood3-knit and knit-fur. Moreover, some images of certain classes were classified in a specific category but not inversely, such as: marble in fur, water in wallpaper and wood1 in fur. Furthermore, there were images in some classes that were misclassified, but it wasn't as frequent as the pairs mentioned before.

3.6. Limitations

One of the main limitatons is that, as the texton dictionary is constructed based on certain group of images, it is hard to find those train images that will be a good descriptor to classify all the test images. Additionally, if the size of the concatenated images is too big, more time is required to filter it and extract the textons. Hence, when the images are too big or there is a large number of classes, a sub-image of each class should be selected to represent its category. Nevertheless, both the size and the sub-section of the image is hard to choose, because the images are selected at random and the small section of some images might not contain representative data of their class.

3.7. Further Work

In order to improve the performance of the algorithm, more images and larger subimages size should be used for the textons construction. However, this would require a better CPU and/or imply more processing time. Moreover, alternative training methods may improve the performance of the algorithm. A neural network with nodes weighting the different texton inputs and 25 outputs (one for each class) can perform similarly and maybe better than the random forest method. Also, training for different image scales can allow to evaluate textures that stand out the most.

4. Conclusions

After using the Ponce group texture database and representing it trough textons, it was possible to train and test two classifiers: k-nearest nighbour and random forest. During the trainning phase, the best combination of parameters was found to be: *number of textons* 175, *number of trees* 75, distance metric standardized Euclidean and a proportion of 70% train and 30% of test images. These parameters were selected because they improved the performance of both classifiers. Finally, they were used in order to test these classifiers resulting in an ACA of 0.73 for KNN and 0.83 for RF. Moreover, it was found that the main classes that caused confusion were wood, floor, knit, marble, granite and bark.

As part of the future work, we would use more and bigger patches of images to create the textons (and features) in order to improve the performance, even though this will

require a better software. Moreover, we would try to implement a neural network that probably will perform even better than RF.

References

[1] K. Chomboon, P. Chujai, P. Teerarassamee, K. Kerdprasop, and N. Kerdprasop. An empirical study of distance metrics for k-nearest neighbor algorithm. In *Proceedings International Conference on Industrial Application Engineering*, pages 280–285, 2015.

[2] R. gate: Jan Ramon University of Leuven. How to determine the number of trees to be generated in random forest algorithm?, 2014.

[3] V. Gholami, K. Chau, F. Fadaee, J. Torkaman, and A. Ghaffari. Modeling of groundwater level fluctuations using dendrochronology in alluvial aquifers. *Journal of hydrology*, 529:1060–1069, 2015.

[4] Z. Guo, Z. Zhang, X. Li, Q. Li, and J. You. Texture classification by texton: Statistical versus binary. *PloS one*, 9(2):e88073, 2014.

[5] 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.

[6] MathWorks. Texture analysis, 2017.

[7] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas. How many trees in a random forest? In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 154–168. Springer, 2012.

[8] D. S. stack exchange. Statistics - train and test data split, 2017.

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

All the codes created and used are attached.