Textons and classifiers laboratory

Diego Martínez
Universidad de Los Andes
Cra 1N 18A-12 Bogota (Colombia)
da.martinez33@uniandes.edu.co

Felipe Torres
Universidad de Los Andes
Cra 1N 18A-12 Bogota (Colombia)
f.torres11@uniandes.edu.co

Abstract

Research in Computer Vision over the years has tried to find a propper way to represent Images according to different descriptors, through research many descriptors have been purposed such as HOG(Histogram of Oriented Gradient)[3], yet one of the oldest descriptors used in the field comes from neuroscience and psychology, according to these areas of knowledge an image can be represented by Texture patterns, then two similar images should have about the same Textures. In this article we created a set of descriptors based on textures found on Ponce's group database for textures[2], and with these descriptors, different classifiers were created to predict the class or category of an image which answered to the descriptors extracted. Classifiers implemented were: Nearest Neighborh and Chi-square metrics for histograms and Random Forest.Then, to evaluate this two methods, confussion matrix and Average Classification Accuracy described the best approach to understand the classification accuraccy. At the end, it was determined that Random forest classifier with 70 trees demonstrate the best classification performance

1. Introduction

Understanding visual information is an easy task for humans thanks to the huge training that our brain has had over the millions of years that we have inhabited this planet, even if we do not think about it too much the centerpiece of our neural network is a fine tuned machine that relies on its many layers in the visual cortex to process images and extract data just by paying attention for a moment; on the other hand this task is a real challenge when it comes to computers and machines that are by far not as nearly trained to do so as our brains.

To begin with, an image can be understood by many ways; for us human it is really easy to find Visual Patterns and rely on them to classify images, yet machines cannot do so in a fast and automatic manner such as we do because for machines an image is just a set of numbers in an array. On the other hand Visual Patterns can be edges, shapes, colors and textures to mention a few of them. Textures in fact are understood by us humans thanks to the neural network present in the Visual cortex in the brain; visual information travels from the eyes to the thalamus and finally to the occipital lobe where in different locations and layers different neurons fire as an answer to an specific texture meanwhile for machines to understand images according to txtures, it is needed first to create a filter bank, then run compute the answers of the image to said filterbank (bear in mind that the answers are computed by convoluting every single time the image with each element of the filter) and finally getting the biggest answers per pixel.

Textures as image descriptors are really rich and powerful but due to the heavy computational power needed to create a Texton dictionary, their usefulness is reduced compared to other Visual Patterns recognised with different methods. For rigid bodies we can find that one of the most useful visual patterns to be used are the edges and shapes, back in 2005 Dalal and Higgs purposed the method *Histogram of Oriented Gradients* that relies on common shapes to identify different classes [3], however this method is not very effective to identify objects whose elements can be deformed (such as articulted bodies), therefore another method based on shapes to classify instances in an image is Deformable part models, which can use shape to describe not only rigid bodies, but several stances of a class in different configurations[4].

2. Methodology

The description of the methodology carried out to represent images using textons in order to train a clasifier of different types of materials, is divided in different stages:

2.1. Database

The database used is the Texture Database from The Ponce Group of Computer Vision and Robotics from Beckman Institute and University of Illinois at Urbana-Champaign [6]. This database is made of 25 texture classes,

with 40 images per class. All images are in grayscale JPG format, 640x480 pixels[2]. To build train and test datasets, 10 images per texture category were selected randomly to construct the test dataset. The remainder images per texture class were used as the database to train the classifier. Various experiments were executed at each stage.

2.2. Image Representation

The representation of database images is base on texture features defined by textons maps, and they were obtain using library of The Berkeley Segmentation Dataset and Benchmark proyect to manipulate textons [1]. First, the filter bank was made of 8 even-symmetric and odd-simmetrics filters defined by the Gaussian second derivative and the Hilbert transform of that Gaussian second derivate respectively, and one center-surround filter at 2 scales. All of this made by ***fbCreate()*** fuction from the bsds library.

After that, the acquirance of dictionary of textons was produced by filtering images with the bank filters, and then clusterizing at a 50 number of centroids, the responses to the filters using ***fbRun()*** and ***computeTextons()*** function from the berkeley library. For this three diferent methodologies were analyzed:

- The dictionary of textons was obtained from 4 images per category of the train database, that were concatenated in a single image, and in the next place, the texton map of each image was determined using ***assignTextons()*** function. This function assign texton maps through squared distances between responses to filter bank and textons of concatenated images.
- The dictionary of textons was obtained from each of 4 images per category of the train database.
- The dictionary of textons was obtained from each of 30 images per category of the train database.

Then, texton histograms were build for each image were constructed using the textons map, these texton histograms were then used as descriptors to be used to train the classifiers.

2.3. Classification

Two different kinds of classifiers were trained using texture representations and annotations of images as histograms:

- Nearest neighbour classifier: which were based in two different metrics that were intersection of histograms and Chi-Square, these two metrics are explained by the following formulaes: For Chi-Square Distance we have

$$d(H_1, H_2) = \sum_{n=1}^{N} \frac{(H_1(I) - H_2(I))^2}{H_1(I)}$$

For Intersection we have:

$$d(H_1, H_2) = \sum_{n=1}^{N} min(H_1(I), H_2(I)) \quad (2)$$

According to these formulaes we decieved a function that works using them as parameters, this function is called *Find_Nearest_Neighborh*. Help command for this function is shown below to have an idea of how to implement it:

```
>> help Find_Nearest_Neighborh
 Function FIND_NEARESt_NEIGHBORH   LABEL = Find_Nearest_Neighborh(HISTDATA,
                                   HISTO,METHOD,LABELS)

                                   Function that finds the nearest
                                   HISTOGRAM neighborh to the one ingressed
                                   as an input. This is done by having a
                                   set of descriptors which are then
                                   checked according to method and the one
                                   with the lowest distance to the input
                                   should be the Nearest Neighborh.
                                   HISTDATA. Set of descriptors to be used
                                   in the comparisson.
                                   HISTO. Histogram to check for it's
                                   nearest neighborh in Histodata.
                                   METHOD. Method to be used when comparing
                                   histograms, it can either be
                                   'Chi-square' or 'Intersection'. By
                                   default it is 'Chi-Square'.
                                   LABELS. Labels containing the names of
                                   each segment of data for the
                                   descriptors, answer will depend on
                                   Labels' names.
```

Figure 1. Help for the implementation of the function *Find_Nearest_Neighborh*

Main function architecture was developed according to OpenCV explanation of histogram comparisson.[5]

Random forest classifier; that was implemented using MATLAB function ***Treebagger()*** based on a classification method described as a parameter of the function, because labels of categories were qualitative and not quantitative, which are used in a regression method. Moreover, the number of trees were variated from 10 to 100 in order to know the incidence of the number of trees in the performance of the classifier

To summarize the experiments implemented:

- 3 different methodologies to obtain the dictionary of textons
- 2 different kinds of classifiers
- 2 different of metrics for Nearest Neighborh classifier
- 10 different number of trees for Random Forest classifier

2.4. Evaluation methodology

Once classifiers were conceived, they were evaluated or tested using category predictions of texture representations from all images of the Test database (10 images per category).

Finally, confussion matrix and Average Classification Accuracy were obtained for each experiment combinations.

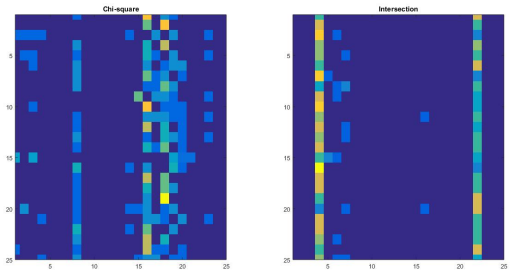

Figure 2. Confusion Matrices for Nearest Neighbor classifier made by the dictionary of 4 per class concatenated images

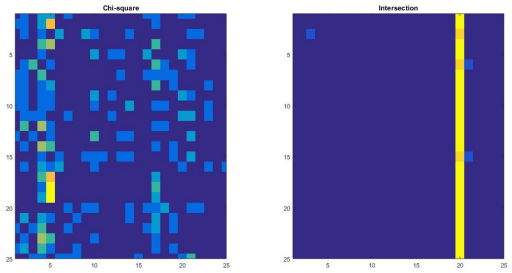

Figure 3. Confusion Matrices for Nearest Neighbor classifier made by the dictionary of 4 per class

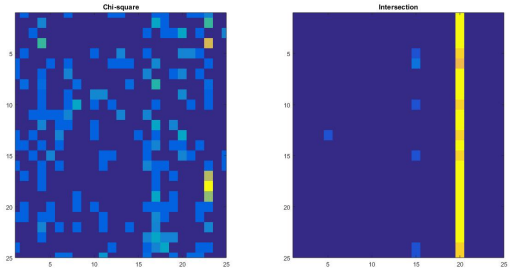

Figure 4. Confusion Matrices for Nearest Neighbor classifier made by the dictionary of all images per class

3. Results

Results of evaluation for Nearest Neighbor classifier trained with the three different metodologies to attain the dictionary of textons are shown at figures 2, 3 and 4.

On the other hand, results of evaluation for Random Forest classifier trained with the three different metodologies to attain the dictionary of textons are shown at figures 5, 6 and 7.

Results of Average Classification Accuracy for Random Forest classifier trained with the three different metodologies to attain the dictionary of textons and at different type of metrics is exhibited in table 1.

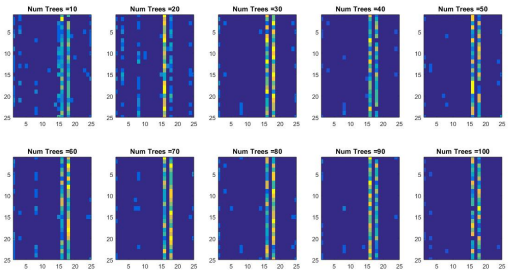

Figure 5. Confusion Matrices for Random forest classifier made by the dictionary of 4 per class concatenated images

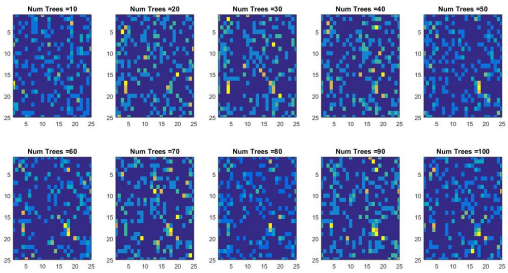

Figure 6. Confusion Matrices for Random forest classifier made by the dictionary of 4 images per class

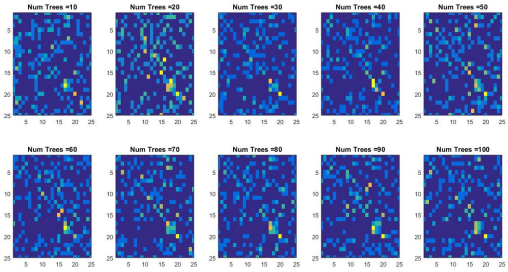

Figure 7. Confusion Matrices for Random forest classifier made by the dictionary of all images per class

Metrics	ACA 1 (%)	ACA 2 (%)	ACA 3 (%)
Chi$_{s}quare$	3,60	3,20	5,34
Intersection	3,60	4,40	4,40

Table 1. ACA results of Nearest Neighbor classifiers for Chi-square and Intersection. ACA 1: Dictionary of 4 per class concatenated images. ACA 2: Dictionary of each 4 per class images. ACA 3: Dictionary of all classes images.

Also, results of Average Classification Accuracy for Random Forest classifier trained with the three different metodologies to attain the dictionary of textons and at a different number of trees is exhibited in table 2.

Trees	ACA 1 (%)	ACA 2 (%)	ACA 3 (%)
10	5,2	6,8	11,2
20	3,6	7,6	10,4
30	4,4	12,0	17,2
40	3,6	10,0	17,2
50	3,6	12,8	15,2
60	6,0	12,0	16,0
70	4,4	10,0	16,8
80	4,4	11,2	14,4
90	3,6	10,4	13,2
100	4,0	12,4	15,2

Table 2. ACA results of Random Forest classifiers with 10 to 100 number of trees. ACA 1: Dictionary of 4 per class concatenated images. ACA 2: Dictionary of each 4 per class images. ACA 3: Dictionary of all classes images.

4. Discussion

Taking into account results of confussion matrices and ACA values for each classifier, it must be said that Random Forest gets better results of preformance that Nearest Neighbor did in all variations of methodologies of classifier training.

In the case of combination of the experiments, the best performance obtained can be infered from table 1. The Random Forest Classifier trained using a texton dictionary made of each image at the Train database and with a trees number of 70, was the classifier showed the higher ACA values, which was of 16.8%.

In addition, if images of confusion matrices are analyzed, values of predictions repetitions or occurences are more significative in some categories that in others. As exposed in figures 2, 3, 4 and 5, categories 16, 18, and 20, which corresponded to *glass1*, *carpet1* and *upholstery*, were the categories that showed a major degree of conusion in the matrices. Other categories as *wood1* and *granite* also had some relevance in confusion intensity at matrices of Figures 2, 3 and 4.

Implementation of Nearest Neighborh classifiers is not a good alternative as could be the Random forest, Nearest Neighborh can be seen as a primitive approach to assign the predicted label to an element in an array; as a response to that the K-Nearest Neighborhs could be better as it does not only take into account the most similar cathegory but also the distances to different categories, thus allowing to have a better ACA than the one obtained in here for Nearest Neighborhs methodology.

As *computeTextons* function was used in the creation of every single entry of the Texton dictionary we can infer that it plays an important role in the performance of the methods, now it is important to mention that his function has an step that relies on the implementation of K-Means, being K the number of textons wanted in the Dictionary, however as we desired to have 50 textons in de dictionary a huge computational effort took place in the computation of K-means with this value, and as the code ran we noticed that it never converged under 100 iterations (MATLAB's default for K-Means), so in order to improve the performance of the code we can either reduce the number of textons or allow K-Means to have more iterations at the cost of being even slower when running the code. (Run time of the slowest experiment was over one and a half day for 750 images).

5. Conlusions

Image representation as taken into account from information obtained by textures allows to create rich descriptors to train classifiers for the images. However, representing images by their textures presents a challenge in computational power due to the need to use K-Means in the creation of the dictionary, rendering a method based on a dictionary of textons only useless in a run on a huge database. Feature extraction for 750 images (25 categories, 30 images per category) takes about one and a half days on a 1T memory and 64 Gb ram Virtual Machine.

Random Forest classifier presents a better alternative to classify images according to textures compared to Nearest Neighborhs, however from the training of the classifiers (3 different ones with different tree values) to the evaluation of the same a 5 core PC will take from half an hour to an hour; meanwhile Nearest neighborhs implementation is way faster, lasting about 10 minutes in a 6 years old machine (less than a minute in newer ones).

To obtain better values of ACA for the experiments listed before, the number of textons in the dictionary can be lowered to make K-means converge in the default iterations, however a different alternative is increasing the number of allowed iterations for the clustering method but making the Texton computation even slower.

References

[1] The Berkeley Segmentation Dataset and Benchmark.

[2] Ponce Research Group: Datasets.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*.

[4] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[5] O. Image Processing. Histograms opencv 2.4.12.0 documentation, 2016.

[6] S. Lazebnik, C. Schmid, and J. Ponce. A Sparse Texture Representation Using Local Affine Regions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1265–1278, Aug. 2005.