Texture classification based on texton histograms

Edgar A. Margffoy-Tuay
Universidad de los Andes
201412566
ea.margffoy10@uniandes.edu.co

Abstract

Texture detection represents historically one of the main approaches to contour and figure segmentation based on the search and detection of repetitive patterns which contrast with other textures present on the image. This problem has been previously addressed by representing textures as a set of feature activations obtained after applying a filter bank over a set of training images, those features, named textons can be used to represent an image and can be used to classify and detect objects and contours. This report pretends to introduce and analyse this framework by implementing a texton-based classifier over the Ponce's group Texture Database.

1. Introduction

A texton is defined as a the minimal unit of human perception and image composition (Analogue to a phoneme in linguistics), this concept proposed by Béla Julesz in 1981, is based on the decomposition of perception as the response to stimulii induced by linear patterns such as intersection segments and crossings. Based on this notion, Malik *et al.* [3] proposed a method in which an image can be represented by the activation responses of each pixel to a set of linear and circular filters, designed to take in account essential visual features such as rotation, scaling, contrast difference and border detection. The geometrical differences between the filters, enables us to discriminate and differentiate different textures based on the patterns that are present, for instance, circular and linear filters allow to distinguish a polka-dot texture from a brick-like one. After grouping each filter activation pixel-wise, it is possible to cluster and quantize each one of these activations (Namely textons) and define a dictionary of textons over a set of images, the texton representation then can be used to describe an image as an histogram that describes the proportion of each class of textons present on the image.

From this approach, it is possible to appreciate that visual objects of the same category may present similar texton representation, due to the geometric and topological similarity between them, this includes similar texture activations in both contour and innings of the objects subject this representation. From this perspective, a classification framework can be proposed in which, images that present a texture label can be represented by its texton histogram that can be related to an specifc category.

2. Materials and Methods

About the Dataset

The Ponce's group texture dataset[1] [2] is a comprehensive image set that contain one texture category (Out of 25 available categories) per sample. Each grayscale image is of size 640×480px, and

[1] Available at `http://www-cvr.ai.uiuc.edu/ponce_grp/data/`

compasses textures like wood, glass, marble, brick, among others. To generate the texton dictionary that describes the images of the dataset, a subset of seven images per class were chosen, also a subset of the original images was randomly selected to conform the image testing set of the texton-based texture classifier.

About the Implementation

To implement the framework proposed on [3], it was necessary to define a set of linear filters, based on Gaussian composition and differentiation, each of the filters (Contrast, edge and circular detectors), was defined on six possible scales and twenty rotations with increasing variances departing from the value 0.1, the scaling was done according to the silver ratio rule ($1 : \sqrt{2}$), this parameter selection allows to capture and select minor local patterns that differentiate similar but different textures categories (*i.e.,* different wood classes). Also increasing the number of filters can be of benefit to adquire more data that can be essential to capture more valid features that can improve the Statistic/Machine Learning classifier accuracy. However, increasing the number of filters may imply an increase on the computational power requierd to process the texton clustering, and therefore the total computing time. To apply the filters over the image set, convolution was applied as a product on Frequency domain, which allows to reduce the time complexity from $\mathcal{O}(N^3)$ to a lower bound of $\mathcal{O}(N \log(N))$, which improves the routine efficiency and the time execution, at the expense of more space consumption.

To describe an image, a texton dictionary of size 128 was used, this result was computed by vector quantization employing K-Means[2]. To improve the accuracy of the clustering result, the KMC[2] centroid seeding algorithm [1] was employed, this routine enables to select the best initial centroids based on a hidden Markov model.

[2]Based on the efficient Mini batched version present on Scikit-Learn library (`http://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html`)

After computing the texton dictionary, it is possible to process all the dataset images as vectors defined on the Texton representation space. To classify the dataset represented as textons features, two classifier models were emplyoing, the first was based on a K-Nearest Neighbors with a chi-squared kernel (1), which allows to compare two different distributions of data, in this case, labels and input texton histograms. The chosen hyperparameter N defined to execute the KNN routine was defined as 10.

$$\chi^2(x, y) = \exp\left(-\sum_i \frac{(x-y)^2}{x+y}\right) \qquad (1)$$

Finally a Random forest model was also trained over the training set, this model employed the gini coeficient to prune the number of branches and cases present on each tree branch, the number of trees was setup as 10 due to the size of the texton-represented dataset, which can tend to Overfitting. Finally, it is necessary to mention that the algorithm routines were implemented on Cython[3], which allows to improve the execution time of the routine.

3. Results

To evaluate the fitness of the model, different filter banks were generated, however, increasing this number lead to more memory consumption which rendered this approach infeasible, in contrast, a modest approach was employed by selecting 79 filters over a subset of 7 images per texture category, all the images were normalized to scale the search space bounds and the representation size from a space of [0, 255]×[0, 255], which may be described by an ellipse to a space of dimension [0, 1]×[0, 1], that is more close to a circle. The process of filter feature computation was accomplished by concatenating the subset of images and multiplying by each of the filters on frequency representation, this approach reduces the total computing time from more than one hour to around thirty minutes in comparison to time

[3]`http://cython.org/`

convolution, consuming around 46Gb on memory resources.

After evaluating the feature response maps, it is possible to conclude that the linear filters contribute more information to the texton representation of the data set, this fact is a consequence of the categories present on the original dataset, in which a great proportion of the images present linear patterns and a reduced proportion are composed mainly of circular features which can improve the differentiation between linear and circular patterns.

To group the pixel activations, an efficient parallel Mini-batched version of K-Means was employed, this choice enables to reduce the total amount of memory required to evaluate this routine, in this case, not all the input data was used to evaluate a single iteration, but a subset. Also, the routine was executed across 16 parallel processors, however this improvement decreased the execution time to one hour and a half. To process the train and test sets, it was necessary to apply the filter bank to each image, due to the number of filters and the size of each image, the process took around 2 hours. From this time, it is possible to conclude that increasing the number of filters increases the time of filter computation, which can be critical if the number of images and its dimensions are more large.

Finally, to train the classifiers, the texton-encoded training images were represented as a columnwise matrix, in which, the ith column contains the ith texton histogram of the ith image of the set. The labels were represented as a $m \times 1$ vector that contains in the ith position the associated numeric label of the ith image. This supervised set was provided to each of the classifiers discused previously, the values of accuracy (2) for each of the models were 33.0231% (KNN) and 26% (Random Forests), this can be accounted to Overfitting due to the small size of the dataset. Also, it is possible to distinguish the models by the set of hyperparameters that govern each of the classifiers, in this case, KNN presents a better adjustment to the test set due to the locallity and the effect of the poll constant N on the model based on the Chi-Squared kernel that is suitable to classify histograms, however, the choice of N can degrade the behaviour of the model. With respect to the Random Forest, it is possible that the branching factor of each of the trees present on the forest implied separating each input image on a leaf, which increases Overfitting.

$$Acc = \frac{||y - \hat{y}||}{||y + \hat{y}||} \tag{2}$$

4. Conclusions

Texton features can extract, resume and characterize an image from the pixel activations observed throughout the image, however, this representation can be complex to generalize and requires many features and groups, which in turn require more computing power. However, this representation is very flexible at the moment to characterize an object that presents relevant texture information, such as textures like wood or marble, however, textures like glass are more difficult to characterize. Also, calculating the filter activations pixelwise can be a computing exhaustive process, which can be improved by using operations in the frequency domain, at the expense of space consumption.

To improve the framework presented throughout this document, different optimizations can be done at compilation time, also, other heuristics can be implemented to improve the memory intensive process of K-means, such as KMC^2, which allows to initialize the best initial guess of centroids available to start the routine. Also, it is necessary to choose wisely the filter set to account to the best contrast and border filters that adjust better to the dataset of interest. However, this filters can be learned by using other Unsupervised Learning techniques, such as Autoencoders or RBMs. Finally, the choice of hyperparameters associated to each one of the classifiers can differentiate the prediction performance, also, the choice of model can be influenced by the datatypes and the

characteristics of the representation chosen for the input variables, for instance, in this case the classification of histograms based on KNN with a Chi-Squared metric is a better solution with respect to a Random Forest.

References

[1] O. Bachem, M. Lucic, S. H. Hassani, and A. Krause. Approximate k-means++ in sublinear time. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 1459–1467. AAAI Press, 2016.

[2] 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.

[3] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43(1):7–27, 2001.