Texture Classification with the PQ Kernel

Radu Tudor Ionescu
Department of Computer Science
University of Bucharest
14 Academiei Street
Romania, Bucharest
raducu.ionescu@gmail.com

Andreea Lavinia Popescu
Politehnica University of Bucharest
313 Splaiul Independentei Street
Romania, Bucharest
andreea.lavinia@ymail.com

Marius Popescu
Department of Computer Science
University of Bucharest
14 Academiei Street
Romania, Bucharest
popescunmarius@gmail.com

ABSTRACT

Computer vision researchers have developed various learning methods based on the bag of words model for image related tasks, including image categorization, image retrieval and texture classification. In this model, images are represented as histograms of visual words (or textons) from a vocabulary that is obtained by clustering local image descriptors. Next, a classifier is trained on the data. Most often, the learning method is a kernel-based one. Various kernels can be plugged in to the kernel method. Popular choices, besides the linear kernel, are the intersection, the Hellinger's, the χ^2 and the Jensen-Shannon kernels. Recent object recognition results indicate that the novel PQ kernel seems to improve the accuracy over most of the state of the art kernels. The PQ kernel is inspired from a set of rank correlation statistics specific for ordinal data, that are based on counting concordant and discordant pairs among two variables. In this paper, the PQ kernel is used for the first time for the task of texture classification. The PQ kernel is computed in $O(n \log n)$ time using an efficient algorithm based on merge sort. The algorithm leverages the use of the PQ kernel for large vocabularies.

Texture classification experiments are conducted to compare the PQ kernel with other state of the art kernels on two benchmark data sets of texture images. The PQ kernel has the best accuracy on both data sets. In terms of time, the PQ kernel becomes comparable with the state of the art Jensen-Shannon kernel. In conclusion, the PQ kernel can be used to obtain a better pairwise similarity between histograms, which, in turn, improves the texture classification accuracy.

Keywords

kernel methods, rank correlation measure, ordinal measure, visual words, bag of words, textons, texture analysis, texture classification.

1 INTRODUCTION

The classical problem in computer vision is that of determining whether or not the image data contains some specific object, feature, or activity. Particular formulations of this problem are object recognition, image classification, texture classification. Computer vision researchers have recently developed sophisticated methods for such image related tasks. Among the state of the art models are discriminative classifiers using the *bag of words* (BOW) representation [ZMLS07, SRE+05] and spatial pyramid matching [LSP06], generative models [FFFP07] or part-based models [LSP05a]. The BOW model, which represents an image as a histogram of local features, has demonstrated impressive levels of performance for image categorization [ZMLS07], image retrieval [PCI+07], texture classification [XZYZ10], or related tasks [IPG13].

One of the early approaches of building a vocabulary of features is [LM01]. The main idea in [LM01] is to construct a vocabulary of prototype tiny surface patches, called 3D textons. Textons obtained by k-means clustering are used for texture classification. *Textons* are elements of texture perception that are widely used in texture analysis. In [XZYZ10], a novel texture classification method via patch-based sparse texton learning is proposed. The dictionary of textons is learned by applying sparse representation to image patches in the training data set. The authors of [VZ05] model textures by the joint distribution of filter responses. This distribution is represented by the frequency histogram of textons.

This work is focused on improving the BOW model by treating texton histograms as ordinal data. In the BOW model, a vocabulary (or codebook) of textons

 ISBN 978-80-86943-71-8

(also termed visual words) is obtained by clustering local image descriptors extracted from images. An image is then represented as a *histogram of textons* (also known as *bag of visual words* or *bag of features*). Next, a classifier is trained on the data. Most often, the learning method is a kernel-based one. Various kernels can be plugged in to the kernel method. Popular choices, besides the linear kernel, are the intersection, the Hellinger's, the χ^2 and the Jensen-Shannon (JS) kernels. All the common kernels used in computer vision treat histograms of textons either as finite probability distributions, either as quantitative random variables whose values are the frequencies of different textons in the respective images. The PQ kernel [IP13] treats the histograms of textons as ordinal data, in which data is ordered but cannot be assumed to have equal distance between values. In this case, a histogram will be interpreted as a ranking of textons according to their frequencies in that histogram. Usage of the ranks of textons instead of the actual values of the frequencies may seem as a loss of information, but the process of ranking can actually make PQ more robust, acting as a filter and eliminating the noise contained in the values of the frequencies.

There is a common perception to avoid rank correlation statistics, such as Kendall's tau [UC04], because they are thought to be computationally expensive. Statements like "the Cayley and Hamming distances are computed in linear time rather than quadratic time like Kendall's tau" from [ZCKB12] can be encountered even in recent publications. Actually, a rank correlation measure based on counting concordant and discordant pairs can be computed in $O(n \log n)$ time based on merge sort [Kni66]. Another algorithm to compute Kendall's tau in $O(n \log n)$ is based on AVL Trees [Chr05]. The author of [Chr05] also ignores the previous work of [Kni66], by stating that: "Traditional algorithms for the calculation of Kendall's tau between two data sets of n samples have a calculation time of $O(n^2)$".

In this work, an efficient algorithm is used to compute the PQ kernel in $O(n \log n)$ time. While the PQ feature map proposed in [IP13] has a quadratic dependence on the number of visual words, the fast algorithm based on merge sort enables the use of the PQ kernel (in the dual form) for large visual word or texton vocabularies.

Recent object class recognition results of [IP13] indicate that the novel PQ kernel seems to improve the accuracy over most of the state of the art kernels. In this paper, texture classification experiments are conducted in order to assess the performance of different kernels, including PQ, on two benchmark data sets of texture images, more precisely, the Brodatz data set and the UIUCTex data set. The empirical results show that the PQ kernel has the best recognition accuracy on both data sets, with a computational time that is less than two times higher than that of the Jensen-Shannon kernel.

The paper is organized as follows. Section 2 presents the bag of visual words model. The PQ kernel for texton histograms is discussed in Section 3. The efficient algorithm to compute the PQ kernel is presented in Section 4. All the experiments are presented in Section 5. Finally, the conclusions are drawn in Section 6.

2 BAG OF VISUAL WORDS MODEL

In computer vision, the BOW model can be applied to image classification and related tasks, by treating image descriptors as words. A bag of visual words is a vector of occurrence counts of a vocabulary of local image features. This representation can also be described as a histogram of visual words. The vocabulary is usually obtained by vector quantizing image features into visual words. In the context of texture classification, visual words are also referred to as textons.

The BOW model can be divided in two major steps. The first step is for feature detection and representation. The second step is to train a kernel method in order to predict the class label of new image. The feature detection and representation step works as follows. Features are detected using a regular grid across the input image. At each interest point, a SIFT feature [Low99] is computed. This approach is known as dense SIFT [DT05, BZM07]. Next, SIFT descriptors are vector quantized into visual words and a vocabulary (or codebook) of visual words is obtained. The vector quantization process is done by k-means clustering [LM01], and visual words are stored in a randomized forest of k-d trees [PCI$^+$07] to reduce search cost. The frequency of each visual word is then recorded in a histogram which represents the final feature vector for the image. The histograms of visual words enter the training step. A kernel method is used for training. Several kernel functions can be used, such as the linear kernel, the intersection kernel, the Hellinger's kernel, the χ^2 kernel, the JS kernel, or the recently introduced PQ kernel [IP13].

It is important to mention that the model described so far ignores spatial relationships between image features. Despite of this fact, visual words showed a high discriminative power and have been used for region or image level classification [CDF$^+$04, FFP05, ZMLS07]. However, the performance can be improved by including spatial information. This can be achieved by dividing the image into spatial bins. The frequency of each visual word is then recorded in a histogram for each bin. The final feature vector of the image is a concatenation of these histograms, which gives a spatial pyramid representation [LSP06] of the image.

 ISBN 978-80-86943-71-8

3 PQ KERNEL

The PQ kernel was introduced in [IP13]. For the sake of completion, the PQ kernel is also presented next. However, it is worth mentioning that the presentation of the PQ kernel provided in this section follows the original presentation given in [IP13].

All common kernels used in computer vision treat histograms either as finite probability distributions, for instance, the Jensen-Shannon kernel, either as quantitative random variables whose values are the frequencies of different textons in the respective images, for instance, the Hellinger's kernel (Bhattacharyya's coefficient) and the χ^2 kernel. Even the linear kernel can be seen as the Pearson's correlation coefficient if the two histograms are standardized. However, the histograms of textons can also be treated as ordinal data, in which data is ordered but cannot be assumed to have equal distance between values. In this case, the values of histograms will be the ranks of visual words according to their frequencies in the image, rather than the actual values of these frequencies.

An entire set of correlation statistics for ordinal data are based on the number of concordant and discordant pairs among two variables. The number of concordant pairs among two variables (or histograms) $X, Y \in \mathbb{R}^n$ is:

$$P = |\{(i,j) : 1 \leq i < j \leq n, (x_i - x_j)(y_i - y_j) > 0\}|.$$

In the same manner, the number of discordant pairs is:

$$Q = |\{(i,j) : 1 \leq i < j \leq n, (x_i - x_j)(y_i - y_j) < 0\}|.$$

Goodman and Kruskal's gamma [UC04] is defined as:

$$\gamma = \frac{P-Q}{P+Q}.$$

Kendall developed several slightly different types of ordinal correlation as alternatives to gamma. *Kendall's tau-a* [UC04] is based on the number of concordant versus discordant pairs, divided by a measure based on the total number of pairs (n is the sample size):

$$\tau_a = \frac{P-Q}{\frac{n(n-1)}{2}}.$$

Kendall's tau-b [UC04] is a similar measure of association based on concordant and discordant pairs, adjusted for the number of ties in ranks. It is calculated as the difference between P and Q divided by the geometric mean of the number of pairs not tied in X and the number of pairs not tied in Y, denoted by X_0 and Y_0, respectively:

$$\tau_b = \frac{P-Q}{\sqrt{(P+Q+X_0)(P+Q+Y_0)}}.$$

All the above three correlation statistics are very related. If n is fixed and X and Y have no ties, then P, X_0 and Y_0 are completely determined by n and Q. Actually, all are based on the difference between P and Q, normalized differently. Following this observation, the PQ kernel between two histograms X and Y is defined as:

$$k_{PQ}(X,Y) = 2(P-Q).$$

The following theorem proves that PQ is indeed a kernel, by showing how to build its feature map.

Theorem 1 *The function denoted by k_{PQ} is a kernel function.*

Proof: To prove that k_{PQ} is a kernel, the explicit feature map induced by k_{PQ} is provided next.

Let $X, Y \in \mathbb{R}^n$ be two histograms of visual words. Let Ψ be defined as follows:

$$\Psi : \mathbb{R}^n \to M_{n,n} \;\; \Psi(X) = (\Psi(X)_{i,j})_{1 \leq i \leq n, 1 \leq j \leq n},$$

with

$$\Psi(X)_{i,j} = \begin{cases} 1 & \text{if } x_i > x_j \\ -1 & \text{if } x_i < x_j \\ 0 & \text{if } x_i = x_j \end{cases}, \forall 1 \leq i,j \leq n.$$

Note that Ψ associates to each histogram a matrix that describes the order of its elements.

If matrices are treated as vectors, then the following equality is true:

$$k_{PQ}(X,Y) = 2(P-Q) = \langle \Psi(X), \Psi(Y) \rangle,$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar product. This proves that k_{PQ} is a kernel and provides the explicit feature map induced by k_{PQ}.

According to [VZ10], the feature vectors of γ-homogeneous kernels should be L_γ-normalized. Being linear in the feature space, PQ is a 2-homogeneous kernel and the feature vectors should be L_2-normalized. Therefore, in the experiments, the PQ kernel is L_2-normalized:

$$\hat{k}_{PQ}(X,Y) = \frac{k_{PQ}(X,Y)}{\sqrt{k_{PQ}(X,X) \cdot k_{PQ}(Y,Y)}}.$$

4 EFFICIENT ALGORITHM FOR PQ KERNEL

It is important to note that for vocabularies with more than 1000 textons, the kernel trick should be employed to directly obtain the PQ kernel matrix instead of computing its feature maps, since there is a quadratic dependence between the size of the feature map and the

number of textons. In general, the kernel trick [STC04] refers to using the dual representation given by the kernel matrix of pairwise similarities between samples instead of the primal representation given by the feature maps. This will greatly reduce the space cost of the PQ kernel. The problem of the time complexity remains to be solved. In the dual form, the PQ kernel between two histograms can be directly computed using an algorithm with a time complexity of $O(n \log n)$, n being the number of textons. The algorithm described in this section follows the work of [Kni66], which proposed a similar algorithm for computing the Kendall's tau measure. The algorithm is based on the key insight of Kendall which, in his book [Ken48], proves that the number of discordant pairs Q between two rankings is equal to the number of interchanges (or swaps), denoted by s, required to transform one ranking into the other. Another important observation is that, given a pair of indices (i, j), interchanging the corresponding values in the same time in the two histograms X and Y, respectively, does not change the number of discordant pairs. Thus, sorting the two histograms X and Y in the same time, using as sorting criteria first the values of X and second (for ties in X) the values of Y, will end up with a new pair of variables X' and Y' that will have the same number of discordant pairs as X and Y. At this point, X' is completely sorted while Y' is not. To represent the same ranking as X', Y' must also be sorted. This implies that the number of swaps needed to sort Y' will be the number of discordant pairs between X' and Y'. As described above, the number of swaps needed to sort Y' is also the number of discordant pairs between X and Y. A slightly modified merge sort algorithm can be used to sort Y' while computing the number s. Algorithm 1 computes k_{PQ} between two histograms $X, Y \in \mathbb{R}^n$ based on these observations.

Algorithm 1: PQ Kernel Algorithm

1. Sort X and Y in the same time using merge sort, according to the values in X in ascending order. If two values in X are equal, sort them according to Y in ascending order.
2. Compute the total number of pairs as $t = n(n-1)/2$.
3. Compute the number of equal pairs in X as e_X .
4. Compute the number of pairs that are equal in both X and Y as $e_{X \wedge Y}$.
5. Sort Y using merge sort in ascending order and sum the differences of swap positions into s.
6. Compute the number of equal pairs in Y as e_Y.
7. Finally, compute $k_{PQ}(X,Y) = 2(t + e_{X \wedge Y} - e_X - e_Y - 2s)$.

Steps 1 and 5 of Algorithm 1 compute the number of discordant pairs Q between X and Y, given that $Q = s$. The number of concordant pairs P is completely determined by the number of discordant pairs, the total numbers of pairs denoted by $t = n(n-1)/2$, the number of equal pairs in X denoted by e_X, the number of equal pairs in Y denoted by e_Y, and the number of pairs that are equal in both X and Y denoted by $e_{X \wedge Y}$. More precisely, P can be expressed as follows:

$$P = t + e_{X \wedge Y} - e_X - e_Y - s.$$

Thus, the difference between P and Q can also be expressed in terms of t, e_X, e_Y, and $e_{X \wedge Y}$, as in step 7 of Algorithm 1.

The analysis of the computational complexity of Algorithm 1 is straightforward. The time for steps 2 and 7 is constant. Steps 1 and 5 are based on merge sort which is known to work in $O(n \log n)$ time. Because X and Y are already sorted, steps 3, 4 and 6 can be computed in linear time with respect to the number of textons n. Consequently, the time complexity of Algorithm 1 is $O(n \log n)$.

5 EXPERIMENTS

5.1 Data Sets Description

Texture classification experiments are presented on two benchmark data sets of texture images. The first experiment is conducted on the Brodatz data set [Bro66]. This data set is probably the best known benchmark used for texture classification, but also one of the most difficult, since it contains 111 classes with only 9 samples per class. Samples of 213×213 pixels are cut using a 3 by 3 grid from larger images of 640×640 pixels. Figure 1 presents three sample images per class of three classes randomly selected from the Brodatz data set.

The second experiment is conducted on the UIUCTex data set of [LSP05b]. It contains 1000 texture images of 640×480 pixels representing different types of textures such as bark, wood, floor, water, and more. There are 25 classes of 40 texture images per class. Textures are viewed under significant scale, viewpoint and illumination changes. Images also include non-rigid deformations. This data set is available for download at `http://www-cvr.ai.uiuc.edu/ponce\_grp`. Figure 2 presents four sample images per class of four classes representing bark, brick, pebbles, and plaid.

5.2 Implementation and Learning Method

Details about the particularities of the learning framework are given next. In the feature detection and representation step, a variant of dense SIFT descriptors extracted at multiple scales is used [BZM07]. The implementation of the BOW model is mostly based on the VLFeat library [VF08].

Various state of the art kernels are compared with the PQ kernel. The kernels proposed for evaluation are

 ISBN 978-80-86943-71-8

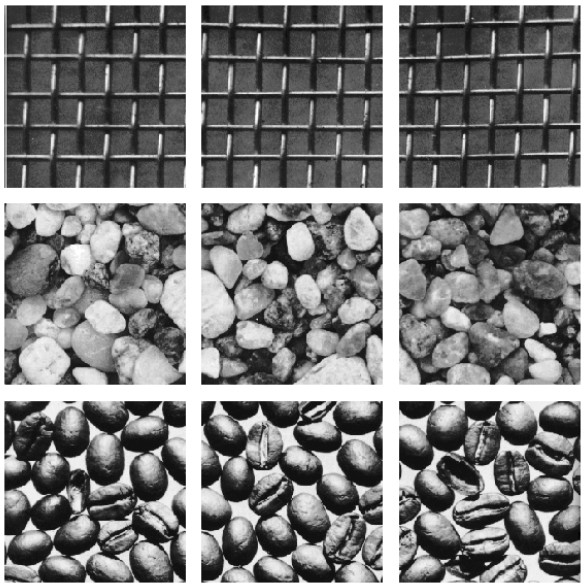

Figure 1: Sample images from three classes of the Brodatz data set.

the L_2-normalized linear kernel, the L_1-normalized Hellinger's kernel, the L_1-normalized intersection kernel, the L_1-normalized Jensen-Shannon kernel, and the L_2-normalized PQ kernel. The norms are chosen according to the authors of [VZ10], that state that γ-homogeneous kernels should be L_γ-normalized. It is important to mention that all these kernels are used in the dual form, that implies using the *kernel trick* to directly build kernel matrices of pairwise similarities between samples.

A state of the art kernel classifier is used in the experiments, namely the Support Vector Machines (SVM), which is very well-suited for binary classification tasks. The SVM [CV95] tries to find the hyperplane that maximally separates the training examples belonging to the two classes. In the experiments, the SVM classifier based on the *one versus all* scheme is used for the multi-class texture classification tasks. More details about the SVM are discussed in [STC04]. The important fact is that the SVM can be trained in such a way that the feature maps are not needed, only the pairwise kernel matrices being required.

5.3 Brodatz Experiment

Preliminary experiments were performed on the Brodatz data set for parameter tuning. First, a set of experiments were conducted to choose the size of the vocabulary. Vocabularies of 1000, 2000, 3000 and 4000 textons were tested, respectively. The best results were obtained with a vocabulary of 2000 textons. Consequently, this vocabulary was selected for the subsequent experiments. The regularization parameter C of the SVM algorithm was chosen by cross-validation for each kernel, on a subset of texture images from the Brodatz data set.

Table 1 compares the accuracy rate of the SVM based on the PQ kernel with the accuracy rates of the SVM based on various state of the art kernels, using 3 random samples per class for training. The accuracy rates presented in Table 1 are actually averages of accuracy rates obtained over 50 runs for each method, in order to reduce accuracy variation induced by the random selection of training and testing samples. The accuracy of the state of the art kernels is well above the accuracy of the baseline linear kernel. More precisely, the state of the art kernels are roughly $2-3\%$ better than the baseline method. However, the BOW model based on the linear kernel achieves comparable results with other state of the art techniques, such as [LSP05b]. In [LSP05b], the accuracy rate reported on the Brodatz data set using the same setup with 3 training samples per class is 88.15%. Among the state of the art kernels, the PQ kernel gives the best accuracy rate. Indeed, the accuracy of the PQ kernel (92.94%) is roughly $0.6-0.8\%$ above the accuracy rates of the other state of the art kernels. The empirical results on the Brodatz data set show that the PQ kernel can outperform the other evaluated kernels for the task of texture classification.

Table 1 also provides the time required by each kernel to produce the kernel matrix that contains pairwise similarities between all the texture samples from Brodatz. Thus, the kernel matrix has 999 rows and 999 columns. The time was measured on a computer with Intel Core i7 2.3 GHz processor and 8 GB of RAM memory using a single Core. Using the efficient algorithm that requires $O(n \log n)$ time to compute the PQ kernel, the

 ISBN 978-80-86943-71-8

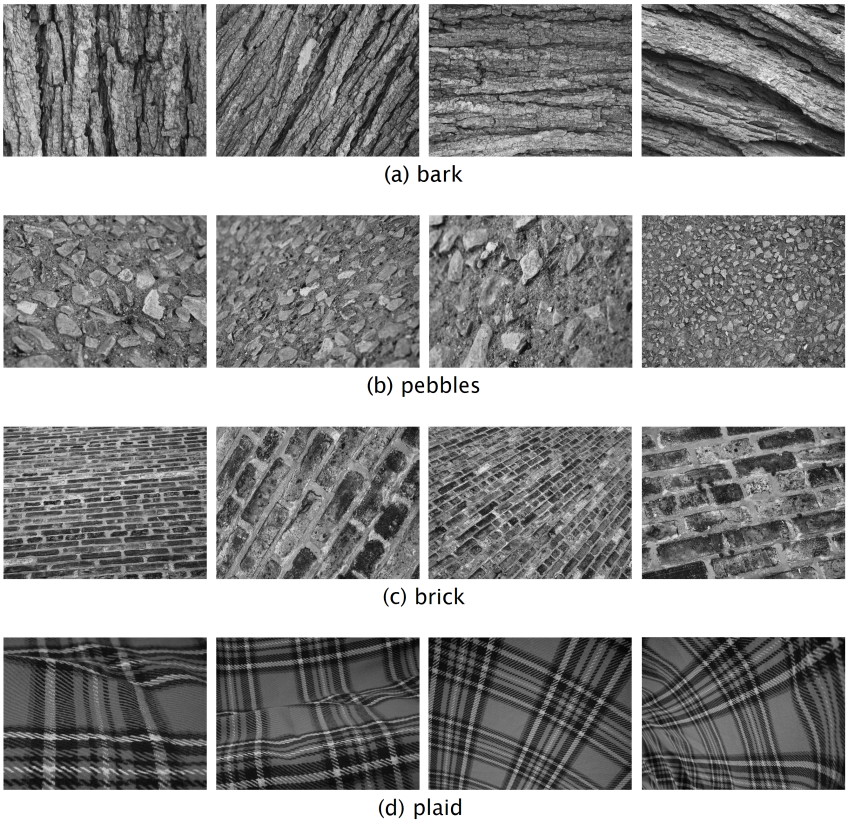

Figure 2: Sample images from four classes of the UIUCTex data set. Each image is showing a textured surface viewed under different poses.

	Lin. L_2	**Hel.** L_1	**Int.** L_1	**JS** L_1	**PQ** L_2
Accuracy	89.63%	92.27%	92.09%	92.33%	92.94%
Time (in seconds)	2.8	3.1	15.9	103.6	151.3

Table 1: Accuracy rates and running times of various kernels on the entire Brodatz data set using 3 random samples per class for training. The PQ kernel is compared with the state of the art kernels on a vocabulary of 2000 textons. The running time required to compute the pairwise similarity matrix for each kernel is reported in this table.

running time of the PQ kernel is now comparable to that of the JS kernel. For a vocabulary of 2000 textons, the PQ kernel is only 1.5 times slower than the JS kernel, which can be computed in $O(n)$ time. The time to compute the quadratic feature maps of the PQ kernel was also measured on the same machine. It takes about 541 seconds to compute the PQ feature maps, thus being almost 3.6 times more computationally expensive than the efficient algorithm for the dual representation of the PQ kernel. While the PQ kernel remains the most computationally expensive kernel among the evaluated kernels, it offers an acceptable trade-off between accuracy and time. In conclusion, Algorithm 1 brings a significant speed improvement and makes the PQ kernel practical for vocabularies of more than 1000 words.

5.4 UIUCTex Experiment

As in the previous case, preliminary experiments were performed on the UIUCTex data set for parameter tuning. Various vocabularies of 1000, 2000, 3000 and 4000 textons were tested, respectively. On this data set, the accuracy seems to improve as the vocabulary dimension is increased. Consequently, the vocabulary of 4000 textons was selected for the subsequent experiments. The regularization parameter C of the SVM algorithm was chosen by cross-validation for each kernel, on a subset of the UIUCTex data set.

Table 2 compares the accuracy rate of the SVM based on the PQ kernel with the accuracy rates of the SVM based on various state of the art kernels, using 20 random samples per class for training. This means that half of the images are used for training, and the other half for testing. The results of the PQ kernel on the UIUCTex data set are consistent with the results obtained on the Brodatz data set. More precisely, the PQ kernel outperforms again the other kernels. The accuracy rate of the PQ kernel is roughly $0.5-0.6\%$ above the accu-

 ISBN 978-80-86943-71-8

	Lin. L_2	**Hel.** L_1	**Int.** L_1	**JS** L_1	**PQ** L_2
Accuracy	84.44%	91.24%	88.22%	91.17%	91.74%
Time (in seconds)	3.4	3.8	18.2	199.8	328.5

Table 2: Accuracy rates and running times of various kernels on the UIUCTex data set using 20 random samples per class for training. The PQ kernel is compared with the state of the art kernels on a vocabulary of 4000 textons. The running time required to compute the pairwise similarity matrix for each kernel is reported in this table.

racy rates of the Hellinger's and the JS kernels. The accuracy rate of the intersection kernel is even lower compared to the PQ kernel. All the state of the art kernels, including the intersection kernel, obtain far better results than the linear kernel. The best accuracy rate of 91.74% is obtained by the PQ kernel. Overall, the empirical results indicate that the PQ kernel can have a better performance for texture classification than the other state of the art kernels.

The time required by each kernel to produce the kernel matrix that contains pairwise similarities between all the texture samples from the UIUCTex data set is also presented in Table 2. In this experiment, the kernel matrix has 1000 rows and 1000 columns, but the number of textons is 4000 this time. The time reported in Table 2 was measured on a computer with Intel Core i7 2.3 GHz processor and 8 GB of RAM memory using a single Core. Again, the PQ kernel remains the most computationally expensive kernel, but the running time of the PQ kernel computed with Algorithm 1 is comparable to the time of the JS kernel. The PQ kernel is roughly 1.6 times slower than the JS kernel. Compared to the Brodatz experiment, the vocabulary is twice as large, containing 4000 textons instead of 2000. However, the proportion between the time of the PQ kernel and the time of the JS kernel does not change by much. The time to compute the quadratic feature maps of the PQ kernel could not be measured on the same machine, since the 8 GB of RAM memory is not enough to store the 1000 feature maps of 16 million features each. This supports the claim that Algorithm 1 makes the PQ kernel practical for large vocabularies, of more than 1000 words. Another fact to support this claim is that the object recognition experiments presented in [IP13] are based on vocabularies of only 500 words, since it is more expensive to compute feature maps of the PQ kernel. More precisely, the feature maps can be computed in $O(n^2)$ time, while the PQ kernel can be computed in the dual form in $O(n \log n)$ time with the efficient algorithm presented in this paper.

6 CONCLUSION AND FURTHER WORK

This paper showed that the PQ kernel can be successfully used for texture classification. An efficient algorithm was used for computing the PQ kernel in the dual form. The algorithm is based on merge sort and needs only $O(n \log n)$ time. The algorithm leverages the use of the PQ kernel in BOW models with large vocabularies. Several texture classification experiments were conducted to show that the PQ kernel can indeed be used with large vocabularies of up to 4000 textons. Futhermore, the empirical results showed that the PQ kernel can improve the BOW model for texture classification in terms of accuracy, by treating texton histograms as ordinal data.

In future work, other methods inspired from ordinal measures can be investigated. For example, the most common correlation statistic for ordinal data, namely the Spearman's rank-order coefficient [UC04], or its L_1-version, namely the Spearman's footrule, are successfully used in text processing [DP09]. It is perfectly reasonable to use them for image categorization in the context of the BOW model, since the BOW model is also inspired from text processing (more precisely, from text retrieval). Methods to transform such ordinal measures into kernels would also have to be developed.

7 ACKNOWLEDGMENTS

The contribution of the authors to this paper is equal. The work of Andreea L. Popescu has been funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/134398.

8 REFERENCES

[Bro66] Phil Brodatz. *Textures: a photographic album for artists and designers*. Dover pictorial archives. Dover Publications, New York, USA, 1966.

[BZM07] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Image Classification using Random Forests and Ferns. *Proceedings of ICCV*, pages 1–8, 2007.

[CDF+04] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and CÃ©dric Bray. Visual categorization with bags of keypoints. *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[Chr05] David Christensen. Fast algorithms for the calculation of Kendall's τ. *Computational Statistics*, 20(1):51–62, 2005.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[DP09] Liviu P. Dinu and Marius Popescu. Comparing Statistical Similarity Measures for Stylistic Multivariate Analysis. *Proceedings of RANLP*, 2009.

[DT05] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. *Proceedings of CVPR*, 1:886–893, 2005.

[FFFP07] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, April 2007.

[FFP05] Li Fei-Fei and Pietro Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. *Proceedings of CVPR*, 2:524–531, 2005.

[IP13] Radu Tudor Ionescu and Marius Popescu. Kernels for Visual Words Histograms. *Proceedings of ICIAP*, 8156:81–90, 2013.

[IPG13] Radu Tudor Ionescu, Marius Popescu, and Cristian Grozea. Local Learning to Improve Bag of Visual Words Model for Facial Expression Recognition. *Workshop on Challenges in Representation Learning, ICML*, 2013.

[Ken48] Maurice G. Kendall. *Rank correlation methods*. Griffin, London, 1948.

[Kni66] William R. Knight. A Computer Method for Calculating Kendall's Tau with Ungrouped Data. *Journal of the American Statistical Association*, 61(314), 1966.

[LM01] Thomas Leung and Jitendra Malik. Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons. *International Journal of Computer Vision*, 43(1):29–44, June 2001.

[Low99] David G. Lowe. Object Recognition from Local Scale-Invariant Features. *Proceedings of ICCV*, 2:1150–1157, 1999.

[LSP05a] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A Maximum Entropy Framework for Part-Based Texture and Object Recognition. *Proceedings of ICCV*, 1:832–838, 2005.

[LSP05b] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A Sparse Texture Representation Using Local Affine Regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, August 2005.

[LSP06] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *Proceedings of CVPR*, 2:2169–2178, 2006.

[PCI+07] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. *Proceedings of CVPR*, pages 1–8, 2007.

[SRE+05] Josef Sivic, Bryan C. Russell, Alexei A. Efros, Andrew Zisserman, and William T. Freeman. Discovering Objects and their Localization in Images. *Proceedings of ICCV*, pages 370–377, 2005.

[STC04] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[UC04] Graham Upton and Ian Cook. *A Dictionary of Statistics*. Oxford University Press, Oxford, 2004.

[VF08] Andrea Vedaldi and B. Fulkerson. VLFeat: An Open and Portable Library of Computer Vision Algorithms. http://www.vlfeat.org/, 2008.

[VZ05] Manik Varma and Andrew Zisserman. A Statistical Approach to Texture Classification from Single Images. *International Journal of Computer Vision*, 62(1-2):61–81, April 2005.

[VZ10] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *Proceedings of CVPR*, pages 3539–3546, 2010.

[XZYZ10] Jin Xie, Lei Zhang, Jane You, and David Zhang. Texture classification via patch-based sparse texton learning. *Proceedings of ICIP*, pages 2737–2740, 2010.

[ZCKB12] Andrew Ziegler, Eric M. Christiansen, David J. Kriegman, and Serge J. Belongie. Locally Uniform Comparison Image Descriptor. *Proceedings of NIPS*, pages 1–9, 2012.

[ZMLS07] Jian Zhang, Marcin Marszalek, Svetlana Lazebnik, and Cordelia Schmid. Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. *International Journal of Computer Vision*, 73(2):213–238, June 2007.