# Speeding Up Local Patch Dissimilarity

Radu Tudor Ionescu and Marius Popescu

Faculty of Mathematics and Computer Science
University of Bucharest, No. 14 Academiei Street, Bucharest, Romania
{raducu.ionescu,popescunmarius}@gmail.com

**Abstract.** There are many patch-based techniques used in image processing, but most of them are heavy to compute with current machines. A dissimilarity measure for images based on patches, inspired from rank distance, called Local Patch Dissimilarity (LPD), was recently introduced. It has very promising results in optical character recognition, but, as other patch-based methods, it is computationally heavy.

This work aims at showing that LPD can be improved in terms of efficiency. Several ways of optimizing the LPD algorithm are presented, such as using a hash table to store precomputed patch distances or skipping the comparison of overlapping patches. Another way to avoid the problem of the higher computational time on large sets of images is to turn to local learning methods.

Several experiments are conducted on two datasets using both standard machine learning methods and local learning methods. All methods are based on LPD. The obtained results come to support the fact that LPD is a very good dissimilarity measure for images. In this paper, LPD is also used with success for classifying images other than handwritten digits.

**Keywords:** image dissimilarity, image classification, handwritten digit recognition, patches, patch-based technique.

## 1 Introduction

Researchers have developed sophisticated methods for analyzing and editing digital images and video. Indeed, many of the most powerful of these methods are patch-based [2] and they divide the image into many small patches and then manipulate or analyze the image based on its patches. Patch-based sampling methods have become a popular tool for image and video synthesis and analysis. Applications include texture synthesis, image and video completion, summarization and retargeting, image recomposition and editing, object detection, noise removal, super-resolution and more.

Local Patch Dissimilarity (LPD) is a dissimilarity measure for images based on patches that was recently introduced in [7]. Patch-based algorithms are known to be computationally heavy because they must search and manipulate millions of patches. Local Patch Dissimilarity is no exception, but it gives a very high accuracy in optical character recognition. This work investigates methods of speeding up the LPD algorithm in order to make it more efficient. Using a hash

A. Petrosino (Ed.): ICIAP 2013, Part I, LNCS 8156, pp. 1–10, 2013.

table to store precomputed patch distances or skipping the comparison step of some patches are the proposed methods for time optimization.

Finding a way of using patch-based algorithms on large datasets or for real-time applications is not always possible. But in some particular cases such as object recognition, image categorization, and image retrieval, there are possible solutions to overcome the time barrier. The aim of this paper is to show that LPD can also be used on large datasets of images. Local learning methods are preferred instead of standard machine learning algorithms to achieve this goal.

The paper is organized as follows. Related work about local learning methods and patch-based techniques is discussed in section 2. LPD is shortly described in section 3. Section 4 presents some optimizations of the LPD algorithm. Experiments with both standard and local learning methods based on LPD are presented in section 5. Finally, the conclusions are drawn in section 6.

## 2 Related Work

Patches belong to the category of local features, which means that they describe properties of a certain region of an image. In contrast to that, global features provide information about an image as a whole. Beside patches, another popular class of local image features are image descriptors, such as SIFT [12].

For numerous computer vision applications, the image can be analyzed at the patch level rather than at the individual pixel level or global level. Patches contain contextual information and have advantages in terms of computation and generalization. For example, patch-based methods produce better results and are much faster than pixel-based methods for texture synthesis [8]. However, patch-based techniques are still heavy to compute with current machines [2].

A paper that describes a patch-based approach for rapid image correlation or template matching is [9]. An approach to object recognition was proposed by [6], where image patches are clustered using the EM algorithm for Gaussian mixture densities and images are represented as histograms of the patches over the (discrete) membership to the clusters. The authors of [1] propose a method where images are represented by binary feature vectors that encode which patches from a codebook appear in the images and which spatial relationship they have. In [17] the authors propose a novel texture classification method via patch-based sparse texton learning. The patch transform, proposed in [4], represents an image as bag of overlapping patches sampled on a regular grid. In [2] the authors present a new randomized algorithm for quickly finding approximate nearest neighbor matches between image patches.

The development of unconventional learning formulations and non-inductive types of inference was studied in [15]. The author argues in favor of introducing and developing unconventional learning methods, as opposed to algorithmic improvements of existing learning methods. This view is consistent with the main principle of VC theory, suggesting that one should always use direct learning formulations for finite sample estimation problems, rather than more general settings (such as density estimation). In [3] the idea of local algorithms for pattern recognition was used, where local linear rules (instead of local constant

rules) and VC bounds (instead of the distance to the $k$-th nearest neighbor) were used. The local linear rules demonstrated an improvement in accuracy on the popular MNIST dataset (from $4,1\%$ to $3,2\%$).

Local learning methods attempt to locally adjust the performance of the training system to the properties of the training set in each area of the input space. A simple local learning algorithm would work like this: for each testing example, select a few training examples located in the vicinity of the testing example, train a classifier with only these few examples and apply the resulting classifier to the testing example. The k-Nearest Neighbor algorithm ($k$-NN) is a method for classifying objects based on the closest training examples in the feature space. It is part of the family of local learning algorithms, but it has the advantage that no training time is required. In this paper, a k-NN with filtering approach is used. For the k-NN with filtering approach, the idea is to filter (or select) the nearest $K$ neighbors (where $K$ is larger than $k$) using a distance measure that is much faster and easy to compute. Instead of training a classifier, the next step is to select the nearest $k$ neighbors from those filtered $K$ examples using a distance measure that is able to capture much finer differences. This latter distance measure is allowed to consume more time in order to determine a better similarity (or dissimilarity) between training examples. This approach is appropriate when it is unreasonable, from the perspective of time, to compute the latter distance for all training examples. This two-step selection (or filtering) process is much faster to compute than a standard k-NN.

## 3 Local Patch Dissimilarity

This section gives an overview of LPD. To compute LPD between two gray-scale images, the idea is to sum up all the offsets of similar patches between the two images. The LPD algorithm is briefly described next. For every patch in one image the idea is to search for a similar patch in the other image. First, look for similar patches in the same position in both images. If those patches are similar, sum up 0 since there is no offset between patches. If the patches are not similar, start looking around the initial patch position in the second image to find a patch similar to the one in the first image. If a similar patch is found during this process, sum up the offset between the two patches. The search goes on until a similar patch is found or until a maximum offset is reached. Note that the maximum patch offset must be set a priori. The time complexity of LPD is $O(h^2 \times w^2)$, where $h$ and $w$ represent minimum height and width of the two compared images, respectively. For a more detailed description of the LPD algorithm and a better understanding of section 4, please refer to [7].

LPD is based on another distance between patches. Any image distance can be used to compute the similarity between patches. Note that the LPD algorithm [7] determines patch similarity using the mean squared euclidean distance that corresponds to the $L_2$-norm. Another version of the LPD algorithm [7] is also used in the experiments, that determines patch similarity using the mean euclidean distance that corresponds to the $L_1$-norm. Both algorithms show good results.

## 4 LPD Algorithm Optimization

As stated in [2], patch-based algorithms are heavy to compute with current computers because these algorithms must deal with millions of patches. Some optimizations that improve the LPD algorithm in terms of speed are discussed here. It may not occur at first look, but LPD needs to compute the similarity between many pairs of patches and for some of them, even several times. Recomputation of patch similarities can be avoided by storing the precomputed values in a hash table. The tests presented in section 5 are performed using the hash table optimization which brings an $8 - 10\%$ speed improvement.

Another optimization is to stop the search for similar patches earlier. Recall that the search goes on until a similar patch is found or until a maximum offset is reached. It is very unlikely to find similar patches at high offset from each other. Even if two similar patches are found at a great distance from each other, it does not mean the images are similar. In fact, this phenomenon may bring noise into the computation of LPD. To avoid this extensive search that can potentially harm the dissimilarity measure, setting a maximum offset radius much lower than the image diagonal size is a good choice. This search limitation was included in all the experiments, which resulted in a great improvement in terms of speed and accuracy. However, one must be careful not to reduce the maximum offset by too much, which can badly alter the performance and strength of the dissimilarity measure. To stop the search too early would mean to disregard some similar patches that bring important information in the dissimilarity computation. In the experiments, an offset radius of $25 - 50\%$ of the image diagonal size works very well. This also brings a speed improvement of $25 - 30\%$.

The last proposed algorithm optimization comes from the fact that LPD computes the similarity between many overlapping patches. A fast version of the LPD algorithm [7] is to skip the comparison of overlapping patches. This can be achieved by increasing the offset by more than one unit, each time a similar patch is not found at the current offset. The results presented in section 5.4 are obtained by increasing the offset with two units at every step, thus, skipping half of the comparisons between patches and speeding the LPD algorithm by a factor of two. The proposed speed improvements do not affect the time complexity.

## 5 Experiments

### 5.1 Datasets

The first dataset used for testing the dissimilarity presented in this paper is the MNIST set, which is described in detail in [11]. Comparative experiments are also reported in [11]. The MNIST database, containing 60.000 training examples and 10.000 test examples, is available at `http://yann.lecun.com/exdb/mnist/`. The second dataset was collected from the Web by the authors of [10] and consists of 100 images each of six different classes of birds: egrets, mandarin ducks, snowy owls, puffins, toucans, and wood ducks. The Birds dataset is used in the last experiment presented in this paper.

**Table 1.** Error and time of 3-NN classifier based on LPD with filtering

| K | Error | Avg. time per image | Overall time |
|---|---|---|---|
| 3 | 2, 73% | 2, 2 seconds | 6 hours |
| 10 | 1, 78% | 2, 6 seconds | 7 hours |
| 30 | 1, 45% | 3, 7 seconds | 10 hours |
| 50 | 1, 38% | 4, 8 seconds | 13 hours |
| 100 | 1, 26% | 7, 6 seconds | 21 hours |
| 200 | 1, 15% | 13, 2 seconds | 36 hours |
| 500 | 1, 09% | 30, 5 seconds | 84 hours |
| 1000 | 1, 05% | 58, 8 seconds | 162 hours |

### 5.2 *k*-NN with Filtering

In [7] LPD was tested on small subsets of the MNIST dataset using a $k$-NN based on LPD model. The obtained error rates are $7,89\%$ on 300 images and $4,59\%$ on 1000 images, improving the baseline $k$-NN error rates by $8,92\%$ and $8,41\%$, respectively. These results look promising, but LPD should be tested on the entire MINST dataset for a strong conclusion. The problem of the $k$-NN classifier based on LPD is that it is not feasible, from a time perspective, for very large datasets. To avoid this problem, local learning methods that speed-up the learning algorithm are used. Therefore, LPD is plugged into different local learning methods and used for image classification.

The first local learning classifier tested here is the $k$-NN with filtering model. This classifier was chosen because it reflects the characteristics of the LPD measure. For the $k$-NN with filtering approach, the idea is to filter the nearest $K$ neighbors using the standard euclidean distance measure (that is much faster to compute). Next, select the nearest $k$ neighbors from those $K$ images using LPD. The two-step selection process is much faster to compute on a large dataset (such as the MNIST dataset) than a standard $k$-NN based entirely on LPD. Results show that this method can improve accuracy and is among the top 4 $k$-NN models that reported results on the MNIST dataset.

The $k$-NN based on LPD with filtering is compared with two other $k$-NN classifiers. One is the $k$-NN based on the euclidean distance measure ($L^2$-norm) between input images. This is the baseline classifier. In [11], an error rate of $5,00\%$ was reported on the regular test set with $k = 3$ for this classifier. Other studies [16] report an error rate of $3,09\%$ on the same experiment. Results obtained in this work also show an error rate of $3,09\%$ on this baseline experiment. The second classifier is the $k$-NN based on Tangent Distance [14]. Tangent distance (TD) is insensitive to small distortions and translations of the input image. The error rate reported for this classifier in [11] is $1,1\%$, but it was necessary to preprocess the images by subsampling to $16 \times 16$ pixels to obtain this error rate.

The accuracy of the $k$-NN based on LPD depends very much on the filtering. Take into account that the nearest $K$ images are selected using the euclidean distance in the filtering phase. If $K$ is close to $k$, a very fast classifier is obtained, but its accuracy will be near the baseline $k$-NN. As $K$ increases the accuracy improves, but the method also becomes slower (since it has to compute LPD between more images than it was before). If $K$ is equal to the number of training

examples, there is no filtering at all and the highest accuracy is obtained. However, the time to compute $k$-NN based on LPD with no filtering is too high to be considered. The idea is to choose an optimal $K$ in order to obtain a trade-off between accuracy and time. Table 1 shows the error rate and the execution time of the 3-NN classifier based on LPD with filtering for several $K$ values. The time was measured on a computer with Intel Core Duo 2.26 GHz processor and 4 GB of RAM memory using a single Core. Empirical results show that an optimal $K$ would be somewhere between 100 and 500. When $K$ is 100, approximately 8 seconds are needed to assign a label to a test image. This is reasonable for a real-time application, since adding parallel processing into assigning a label can improve the time even further.

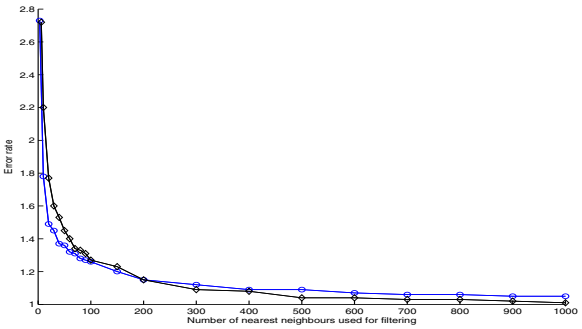


**Fig. 1.** Error rate drops as K increases for 3-NN (○) and 6-NN (◇) classifiers based on LPD with filtering

Looking at how the error gets lower as $K$ increases, one can observe that the error tends to stabilize at some point. In other words, the error rate will not drop anymore after a certain $K$ value. That error rate represents the actual error rate of a 3-NN classifier based on LPD (without filtering). The limitation of LPD induces this error, but what exactly is this error rate? Figure 1 gives an overview of this phenomenon and a hint about the point where the error stabilizes for both 3-NN and 6-NN classifiers. In order to make a prediction about the stabilization point of the error, the stability of the $k$-NN with filtering needs to be studied as $K$ varies. It is clear from Figure 1 that the error drops with no variation when $K$ increases. But increasing $K$ may induce a misclassification on some test images (even if, overall, more images are classified correctly). For $K = 100, 200, ..., 1000$, the set of misclassified images for a certain value of $K$ always includes the set of misclassified images for a greater value of $K$. Thus, the method has very stable behavior as $K$ varies. In these circumstances, the error rate of the 3-NN and 6-NN classifiers based on LPD (without filtering) can be obtained by testing it only on the previously misclassified images. Doing so, an error rate of $1,03\%$ is obtained for the 3-NN classifier and an error rate of $0,98\%$ is obtained for the 6-NN classifier. These error rates are based only on a statistical proof. But as stated before, the real proof (that of testing the 3-NN and 6-NN classifiers with no filtering on all test images) is not practical from the time perspective.

**Table 2.** Error on MNIST dataset for baseline 3-NN, $k$-NN based on tangent distance (TD) and $k$-NN based on LPD with filtering

| **Method** | baseline 3-NN | $k$-NN + TD | 3-NN + LPD + Filter | 6-NN + LPD + Filter |
|---|---|---|---|---|
| **Error** | $3,09\%$ | $1,1\%$ | $1,05\%$ | $\mathbf{1,01}\%$ |

The obtained error rates only give a good indication of the actual ones, and they are not compared with the error rates of the other $k$-NN classifiers based on euclidean and tangent distance, respectively.

Table 2 compares error rates of the three $k$-NN classification methods (distinct only by the metric used) using the MNIST test set of 10.000 examples. For the $k$-NN classifier based on LPD with filtering, $k = 3$ and $k = 6$ are used. The reported results use the nearest $K = 1000$ images filtered with the euclidean distance. Note that the error rates of the $k$-NN models based on LPD are obtained with patches of $4 \times 4$ pixels and a similarity threshold of $0,125$. These parameters are obtain through 10-fold cross validation on 1000 images selected from the MNIST dataset. The $k$-NN based on LPD with filtering model has a better accuracy than the other $k$-NN models. In fact, it is among the top 4 $k$-NN models that reported results on the MNIST dataset. The 6-NN classifier based on LPD with filtering has an error rate of only $1,01\%$. Note that unlike the $k$-NN with filtering based on LPD, the other three methods from top 4 need additional preprocessing steps.

### 5.3 Local Learning

The idea of combining LPD with kernel methods (such as SVM or kernel Ridge Regression) was presented in [7]. The reported error rates for the SVM based on LPD are $5,67\%$ on 300 images and $3,10\%$ on 1000 images, improving the standard SVM error rates by $8,00\%$ and $5,60\%$, respectively. The kernel Ridge Regression (kRR) based on LPD has an error rate of $6,96\%$ on 300 images and an error rate of $3,84\%$ on 1000 images, improving again the standard kRR error rates by $6,85\%$ and $4,58\%$, respectively. The accuracy improves when there are more training samples.

Kernel methods are based on similarity. LPD can be transformed into a similarity measure using the Gaussian-like kernel [13]:

$$K_{LPD}(img^1, img^2) = e^{-\frac{LPD(img^1, img^2)}{2\sigma^2}},$$

where $img^1$ and $img^2$ are two gray-scale images. The parameter $\sigma$ is usually chosen to match the number of features (pixels) so that values of $K_{LPD}$ are well scaled. Because LPD is computationally heavy, it is not feasible to compute a kernel matrix with high dimensions. Therefore, classifiers such as SVM and kRR based on LPD cannot be used directly on the entire MINST dataset, for example. Instead, kernel methods can be integrated into a local learning algorithm. The proposed approach is very similar to the $k$-NN with filtering model. The first step is to filter the nearest $K$ neighbors using the standard euclidean distance. The second step is to train a kernel classifier using only the filtered $K$ neighbors.

A new classifier will be trained for each test image. Each classifier will predict only the label of the test image that was build for.

Before training the classifier, it is necessary to build a kernel matrix using LPD. It is not feasible for a real-time application to build a $K \times K$ kernel matrix for each test image, when $K$ is larger than 100. But, a large number of $K$ neighbors is needed in order to improve the accuracy of the kernel method. A feasible solution is to train a kernel classifier only when there is not a majority of images with the same label greater than 60% among the filtered $K$ neighbors, and use a $k$-NN model otherwise. There are two reasons for this approach. First, if there is such a majority, the kernel method will be biased towards choosing the majority class. Second, it is likely that a simple $k$-NN would also choose this majority class that is probably the right one.

This local learning algorithm was tested on the MNIST dataset. Using $K = 200$, it gives an error rate of $1,07\%$. From the 10.000 test cases, 983 of them had a majority class of less than 60% of the total 200 neighbors. For each of these 983 cases, a kRR classifier based on LPD was trained. The other test labels were predicted using the 3-NN based on LPD. Note that the 3-NN with filtering approach has an error rate of $1,15\%$ for $K = 200$ and the local learning algorithm is able to improve it (to $1,07\%$). Another local learning algorithm, that uses the SVM classifier instead of kRR classifier, was tested without being able to improve the accuracy.

In conclusion, using kernel methods in a local learning context doesn't bring a significant improvement in accuracy to the $k$-NN with filtering approach. However, the local learning algorithm proposed here can be successfully used for handwritten digit recognition. It also benefits from a faster computational time compared to standard kernel methods based on LPD.

### 5.4 Skip Overlapping Patches

In this experiment, LPD is used to classify a more general type of images available in the Birds dataset. Several $k$-NN models based on different distance measures are compared. The first $k$-NN model uses the Bhattacharyya coefficient to compare spatiograms of HSV values extracted from images. This improved measure of comparing spatiograms is proposed in [5]. The second $k$-NN model is based on the mean euclidean distance measure ($L_1$-norm) between input images. Another two $k$-NN classifiers based on LPD are tested. One of them uses a slightly modified version of the LPD algorithm [7]. It determines similar patches using the mean euclidean distance corresponding to the $L_1$-norm. The other one uses a fast version of the LPD algorithm that skips half of the comparisons between patches.

For both the euclidean distance measure and the LPD measure, images from the Birds dataset need to have the same size in order to be compared. Thus, images are resampled to $100 \times 100$ pixels. To observe the difference between original and resampled images, the $k$-NN model based on the Bhattacharyya coefficient is tested on both types of images. The other $k$-NN models are tested only on the resampled images, that are also converted to gray-scale. Table 3 compares error rates of all $k$-NN models.

**Table 3.** Error on Birds dataset for different $k$-NN models

| **Method** | **Preprocessing** | **Error** |
|---|---|---|
| 5-NN + Bhattacharyya | none | $57,33\%$ |
| 5-NN + Bhattacharyya | resize | $54,67\%$ |
| 3-NN + euclidean | resize, gray-scale | $51,00\%$ |
| 3-NN + LPD | resize, gray-scale | $\mathbf{30,33}\%$ |
| 3-NN + LPD + skip | resize, gray-scale | $\mathbf{30,33}\%$ |

**Table 4.** Error on Birds dataset for texton learning methods and kernel methods based on LPD

| **Method** | Naive Bayes | Exp+parts | Exp+rel | Exp+parts+rel | SVM+LPD+skip | kRR+LPD+skip |
|---|---|---|---|---|---|---|
| **Error** | $21,33\%$ | $\mathbf{7,67}\%$ | $24,67\%$ | $8,33\%$ | $27,33\%$ | $26,00\%$ |

Next, two kernel methods based on the fast version of LPD (that skips half of the comparisons between patches) are compared with the state of the art texton learning methods from [10]. The proposed kernel methods based on LPD are the SVM and the kernel Ridge Regression. Table 4 compares error rates of kernel methods based on LPD and texton learning methods. The kernel methods based on LPD have an accuracy similar to some of the state of the art methods. However, the proposed kernel methods are more time efficient. The authors of [10] report a time of about 7 days for a single experiment, while the kernel methods based on LPD need about 4 days for the same experiment on an Intel Core Duo 2.26 GHz processor and 4 GB of RAM memory. Note that error rates of all models based on LPD, used in this experiment, are obtained with patches of $4 \times 4$ pixels and a similarity threshold of $0,15$. In conclusion, the fast version of LPD can successfully be used as a kernel for image classification.

## 6 Conclusion and Further Work

This work presented several methods of speeding up the LPD algorithm, showing that LPD can be used for real-time image classification, especially when local learning methods are preferred instead of standard machine learning algorithms. Local learning methods based on LPD were proposed and tested on the popular MNIST dataset. The experiments show that local learning algorithms perform very well in terms of accuracy and time. The error rate achieved by the $k$-NN based on LPD with filtering approach is only $1,01\%$ on the MNIST dataset.

Another important achievement of this work is that LPD has successfully been used for classifying images other than handwritten digits. The kRR based on LPD obtained an error rate of $26,00\%$ on the Birds dataset, which is comparable to the results obtained in [10]. In conclusion, LPD is a powerful dissimilarity measure that can be used with good results in important problems in image processing, such as object classification, object recognition or similar tasks.

In future work, a very fast version of LPD based on image descriptors can be proposed. Instead of comparing the similarity between many patches, LPD can compare the similarity of a few SIFT descriptors, for example. This approach

would bring a major improvement in terms of speed. Until further investigation, it remains uncertain if such an approach can have similar or more accurate results than the current formulation of LPD. A hint could be that LPD somehow measures the difference of spatial information between images. Can this difference still be measured using fewer image descriptors?

## References

1. Agarwal, S., Roth, D.: Learning a sparse representation for object detection. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002, Part IV. LNCS, vol. 2353, pp. 113–127. Springer, Heidelberg (2002)
2. Barnes, C., Goldman, D.B., Shechtman, E., Finkelstein, A.: The PatchMatch Randomized Matching Algorithm for Image Manipulation. Communications of the ACM 54(11), 103–110 (2011)
3. Bottou, L., Vapnik, V.: Local Learning Algorithms. Neural Computation 4, 888–900 (1992)
4. Cho, T.S., Avidan, S., Freeman, W.T.: The patch transform. PAMI 32(8), 1489–1501 (2010)
5. Conaire, C.O., O'Connor, N.E., Smeaton, A.F.: An Improved Spatiogram Similarity Measure for Robust Object Localisation. In: ICASSP, pp. 1069–1072 (2007)
6. Deselaers, T., Keyser, D., Ney, H.: Discriminative Training for Object Recognition using Image Patches. In: CVPR, pp. 157–162 (2005)
7. Dinu, L.P., Ionescu, R.-T., Popescu, M.: Local patch dissimilarity for images. In: Huang, T., Zeng, Z., Li, C., Leung, C.S. (eds.) ICONIP 2012, Part I. LNCS, vol. 7663, pp. 117–126. Springer, Heidelberg (2012)
8. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: SIGGRAPH 2001, pp. 341–346. ACM (2001)
9. Guo, G., Dyer, C.R.: Patch-based Image Correlation with Rapid Filtering. In: CVPR (2007)
10. Lazebnik, S., Schmid, C., Ponce, J.: A Maximum Entropy Framework for Part-Based Texture and Object Recognition. In: ICCV, pp. 832–838 (2005)
11. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
12. Lowe, D.G.: Object Recognition from Local Scale-Invariant Features. In: ICCV, vol. 2, pp. 1150–1157. IEEE Computer Society, Washington, DC (1999)
13. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
14. Simard, P., LeCun, Y., Denker, J.S., Victorri, B.: Transformation Invariance in Pattern Recognition, Tangent Distance and Tangent Propagation. Neural Networks: Tricks of the Trade (1996)
15. Vapnik, V.: Estimation of dependencies based on empirical data (Information Science and Statistics), 2nd edn. Springer (2006)
16. Wilder, K.J.: Decision tree algorithms for handwritten digit recognition. Electronic Doctoral Dissertations for UMass Amherst (January 1998), `http://scholarworks.umass.edu/dissertations/AAI9823791`
17. Xie, J., Zhang, L., You, J., Zhang, D.: Texture classification via patch-based sparse texton learning. In: ICIP, Hong Kong, China, pp. 2737–2740 (2010)