Face Recognition with Optimized Tree-Structured Local Binary Patterns

Daniel Maturana, Domingo Mery and Álvaro Soto
Departamento de Ciencias de la Computación
Pontificia Universidad Católica
Santiago, Chile
Email: {dimatura, dmery, asoto}@uc.cl

***Abstract*—Many state-of-the-art face recognition algorithms use image descriptors based on features known as Local Binary Patterns (LBPs). In this work we describe a method to learn LBP-like descriptors that learn the most discriminative features for each facial region in a supervised manner. The method represents a set of pixel comparisons as a tree and searches for a tree which maximizes a Fisher-based class separability criterion for the descriptors. The optimization is performed heuristically by stochastic hill climbing. Tests on standard face recognition datasets show the method creates highly discriminative yet compact descriptors.**

***Keywords*-face recognition; local binary patterns; decision tree; id3; hill climbing; nearest neighbor;**

I. Introduction

While face recognition algorithms commonly assume that face images are well aligned and have a similar pose, in many practical applications it is impossible to meet these conditions. Therefore extending face recognition to less constrained face images has become an active area of research.

To this end, face recognition algorithms based on properties of small regions of face images – often known as local appearance descriptors or simply local descriptors – have shown excellent performance on standard face recognition datasets. Examples include the use Gabor features [1], SURF [2], SIFT [3], HOG [4], and histograms of Local Binary Patterns (LBPs) [5]. A comparison of various local descriptor-based face recognition algorithms may be found in Ruiz del Solar et al [6].

Among the different local descriptors in the literature, histograms of LBPs [7] have become popular for face recognition tasks due to their simplicity, computational efficiency, and robustness to changes in illumination. The success of LBPs has inspired several variations. These include local ternary patterns [8], elongated local binary patterns [9], multi scale LBPs [10], patch based LBPs [11], LBPs on Gabor magnitude images [12], to cite a few. However, these are specified a priori without any input from the data itself, except in the form of cross-validation to set parameters.

In this paper, our main contribution is to propose a new method that explicitly learns discriminative descriptors from the training data. This method builds on and improves on our previous work [13] by modifying the descriptor learning stage and adding weights to the nearest neighbor classification. As a testing scenario, we consider the traditional task of *closed set face identification*. Under this task, we are given a gallery of identified face images, such that, for any unidentified probe image, the goal is to return one of the identities from the gallery.

This paper is organized as follows. Section II presents general background information about the operation of traditional LBPs and also about the pipeline used by our approach to achieve face recognition. Section III presents the main details of our approach. Section IV discusses relevant previous work. Section V shows the main experiments and results of applying our approach to two standard benchmark datasets. Finally, Section VI presents the main conclusions of this work.

II. Background Information

A. Local Binary Patterns

Local binary patterns were introduced by Ojala et al [7] as a fine scale texture descriptor. In its simplest form, an LBP description of a pixel is created by thresholding the values of a 3×3 neighborhood with respect its central pixel and interpreting the result as a binary number.

In a more general setting, a LBP operator assigns a decimal number to a pair $(c, \mathbf{n})$,

$$b = \sum_{i=1}^{s} 2^{i-1} I(c, n_i)$$

where c represents a center pixels and $\mathbf{n} = (n_1, \dots n_s)$ corresponds to a set of pixels sampled from the neighborhood of c according to a given pattern. Also,

$$I(c, n_i) = \begin{cases} 1 & \text{if } c \leq n_i \\ 0 & \text{otherwise} \end{cases}$$

This can be seen as assigning a 0 to each neighbor pixel in $\mathbf{n}$ that is larger than the center pixel c, a 1 to each neighbor smaller than c, and interpreting the result as a number in base 2. In this way, for the case of a neighborhood of s pixels, there are 2^s possible LBP values.

B. Face recognition pipeline

Our face recognition pipeline is similar to the one proposed in [5], but adds a more sophisticated illumination normalization step, proposed by Tan and Triggs [8]. Figure (1) summarizes its operation, given by the following main steps:

1) Crop the face region and align the face by mapping the eyes to a canonical location with a similarity transform.
2) Normalize illumination with Tan and Triggs' [8] Difference of Gaussians filter.
3) Partition the face image in a grid with equally sized cells, the size of which is a parameter.
4) For each grid cell, apply a feature extraction operator (such as LBPs) to each pixel in the grid cell. Afterward, create a histogram H_i of the feature values and concatenate these histograms into a single "spatial histogram" $S = (H_1, \cdots, H_M)$.
5) Classify a probe face with the identity of the nearest neighbor in the gallery, where the nearest neighbor distance is calculated with the (possibly weighted) L_1 distance between the histograms of the corresponding face images. In our algorithm, we use one weight for each grid cell. That is, the distance between two spatial histograms $S^1 = (H_1^1, \cdots, H_M^1)$ and $S^2 = (H_1^2, \cdots H_M^2)$ is

$$dist(S^1, S^2) = \sum_{m=1}^{M} w_m \sum_i |H_{mi}^1 - H_{mi}^2| \quad (1)$$

where H_{mi} is the ith bin of the mth histogram. We specify how the weights are obtained below.

III. Our Approach: Optimized Tree-Structured Local Binary Patterns

The simple observation behind OT-LBP is that the operation of a LBP over a given neighborhood is equivalent to the application of a fixed binary decision tree [13]. In effect, the aforementioned histograms of LBPs may be seen as quantizing each pair $(c, \mathbf{n})$ with a specially constructed binary decision tree, where each possible branch of the tree encodes a particular LBP. The tree has s levels, where all the nodes at a generic level l compare the center pixel c with a given neighbor $n_l \in \mathbf{n}$. In this way, at each level $l-1$, the decision is such that, if $c < n_l$ the vector is assigned to the left node; otherwise, it is assigned to the right node. Since the tree is complete, at level 0 we have 2^s leaf nodes. Each of these nodes corresponds to one of the 2^s possible LBPs. In fact, seen as a binary number, each LBP encodes the path taken by $(c, \mathbf{n})$ through the tree; for example, in a LBP with $s = 8$, `11111101` corresponds to a $(c, \mathbf{n})$ pair which has taken the left path at level $l = 1$ and taken the right path at all other levels.

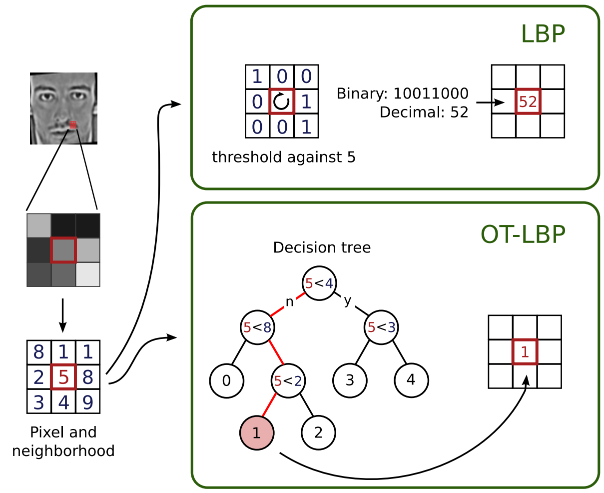


Figure 2. The LBP operator versus the OT-LBP operator.

The previous equivalence suggests the possibility of using a different tree construction scheme to learn discriminative LBP-like descriptors from training data. One possibility is to use a standard ID3-style learning algorithm [13], [14]. We have found that we can obtain significative accuracy improvements on this by learning an initial tree with the standard ID3-style algorithm and then further optimizing it with regards to a Fisher-based class separability criterion by stochastic hill climbing. We dub this approach *Optimized Tree-Structured Local Binary Patterns* or OT-LBP. As a major advantage, by using training data to learn the structure of the tree, OT-LBP can effectively build an adaptive tree, whose main branches are specially tuned to encode discriminative patterns for the relevant target class.

Figure 2 illustrates the operation of regular LBPs and OT-LBPs. After a decision tree is trained, OT-LBP assigns to each leaf node a code given by the path or branch that leads to that node in the tree. In this way, for any input pixel c and the corresponding neighborhood $\mathbf{n}$ used to build the tree, the pair $(c, \mathbf{n})$ moves down the tree according to the $c < n_l$ comparisons. Once it reaches a leaf node, the respective code is assigned to the center pixel c (code number 1 in Figure 2). As with ordinary LBPs, the OT-LBPs obtained for a given image can be used for classification by building histograms.

In summary the proposed approach has the following advantages:

- We use the data to explicitly and automatically search for discriminative LBPs instead of handcrafting the structure of the descriptor.
- Since we expect different patterns to be discriminative in different face image regions, we learn a different tree for each region.
- Instead of neighborhood of eight or sixteen pixels as in regular LPBs, we use a much larger neighborhood

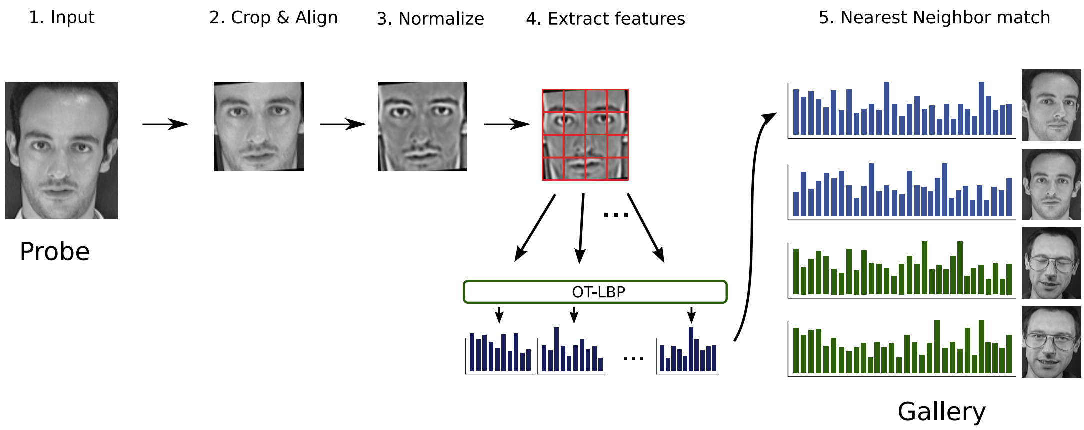


Figure 1. The face recognition pipeline.

and let the tree construction algorithm decide which neighbors are more relevant.
- Apart from the feature extraction step, OT-LBP can be used with no modification in any of the many applications where LBP is currently applied.

A. *Tree construction*

Our objective is to construct a tree that will quantize the pixels into histograms with a large within-subject similarity and low between-subject similarity. After testing various criteria to quantify this objective, we have found the following Fisher-like criterion to work best:

$$J = \frac{(\mu_w - \mu_b)^2}{\sigma_w^2 + \sigma_b^2} \tag{2}$$

where μ_w and μ_b are the mean within-subject and between-subject distances of the histograms induced by the tree, and σ_w^2 and σ_b^2 are the variances of the within-subject and between-subject distances of the histograms induced by the tree. This criterion was used by Zhang et al [12] to weigh different facial regions.

In practice we have found that larger trees are more discriminative, but this has a cost in terms of storage space and computational efficiency. Therefore wish to control the size of the tree (and thus, the histogram size) as a parameter. Finding the tree that optimizes (2) subject to a size constraint is extremely complex, and we resort to a heuristic procedure.

Our initial attempts used Genetic Programming (GP) [15], which uses a evolutionary search directly in the space of trees. However, we found it difficult to combat bloat, the well known-tendency of GP to favor ever-larger trees, while mantaining a precise control of tree size.

We found a much simpler scheme to work better in terms of solution quality and computational cost. The scheme has two stages. In the first stage, a good initial tree is is grown using a simple version of Quinlan's greedy ID3 algorithm [14] to induce the trees. The trees are recursively built top-down by splitting the pairs $(c, \mathbf{n})$ in each node with a decision of the form $c < n_i$, with $1 \leq i \leq s$. The decision is chosen greedily at each node via the usual information gain criterion.

$$\arg\max_i \Delta H(n_i) = H_n - p_r H_r - p_l H_l,$$

where H_n is the entropy of the current node, H_r and H_l are the entropies of the left and right nodes induced by the decision n_i. Likewise, p_r and p_l are the proportions of $(c, \mathbf{n})$ pairs that go to the left and right nodes according to the decision. This criterion favors splits that discriminate between pairs from each class. The data is split until the maximum specified depth is reached.

Once the tree is fully grown, we begin a second stage in which we directly optimize the original criterion (2) by iteratively improving the initial tree. The improvement is performed by stochastic hill climbing. In pseudo code, this is:

```
proc hillclimb(tree) ≡
  J* := −inf
  for i := 1 to hill_climbing_iterations do
      new_tree := copy(tree)
      J' := J(new_tree)
      for j := 1 to tweaks do
          tree' := tweak(copy(tree))
          if J(tree') ≥ J'
            then
                J' := J(tree')
                new_tree := tree'
          fi
      od
      tree := new_tree
      if J' ≥ J*
        then
            best := tree
            J* := J'
      fi
```

od
return **best**

.

proc J(tree) $\equiv$
Quantize data into histograms with **tree**
evaluate and return $(\mu_w - \mu_b)^2 / \left(\sigma_w^2 + \sigma_b^2\right)$

.

proc tweak(tree) $\equiv$
Select random non-leaf node from **tree**
Set the n_i in $(n_i < c)$ to a random integer a in $(1, s)$,
s.t. none of the ancestors or descendants of the node
test for $(a < c)$
return **tree**

.

Finally, we keep the best tree obtained during the hill climbing phase and its associated J^* value. As a further refinement with respect to [13], we use this value as a weight for the distance calculation in (1), assuming that J^* reflects how discriminative the facial region corresponding the grid cell is.

We define neighborhood $\mathbf{n}$ used by OT-LBP differently than LBPs. We use a square neighborhood centered around c, and instead of samples taken along a circle, as in regular LBPs, we consider all pixels inside the square as part of the neighborhood (fig. (3)). All the pixels within this square are considered as potential split candidates. The idea is to let the tree construction algorithm find the most discriminative pixel comparisons.

The main parameters of this algorithm are the size of the neighborhood $\mathbf{n}$ to explore, the maximum depth of the trees, the number of hill climbing iterations and the number of tweaks tested at each hill climbing iteration. As shown in Figure 3, the first parameter is determined by a radius r. We have observed larger r values results in better performing trees, but are more expensive in the first phase of the tree construction process. We empirically set $r = 7$ as a compromise. The second parameter, tree depth, determines the size of the resulting histograms. Smaller histograms are desirable for space and time efficiency, but as we mentioned, there is a trade-off in accuracy with respect to larger histograms. We will report accuracies with depths between 6 and 9. With regard to the last two parameters, these are dictated mostly by the available computing resources. In our C++ implementation, 100 hill climbing iterations and 30 tweaks tested per iteration keeps the training to take around a day for trees with maximum depth 9.

Using trees opens up various possibilities. We have explored some extensions to the basic idea, such as using a forest of randomized trees (as in [16] and [17]), trees splitting based on a linear combinations of the values of the neighborhood (i.e. nodes split on $\mathbf{n}^T\mathbf{w} < c$, similarly to [18]), or using ternary trees where a middle branch corresponds to pairs for which $|c - n_i| < \epsilon$. This last approach can be considered as the tree-based version of the "Local Ternary Pattern" in [8]. So far, we found that a single tree built with our scheme is the best performing solution.

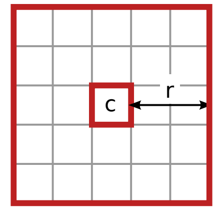


Figure 3. Pixel neighborhood used in OTLBP. The inner square is the center pixel c, and the neighborhood corresponds to all the pixels enclosed in the larger square. The size of the neighborhood is determined by the radius r.

IV. Related work

Our algorithm can be seen as a way to quantize $(c, \mathbf{n})$ pairs using a codebook where each code corresponds to a leaf node. This links our algorithm to various other works in vision that use codebooks of image features to describe the images.

Ahonen et al [19] proposed to view the difference $c - n_i$ of each neighbor pixel n_i with the center as the response of a particular filter centered on c. Under this view, the LBP operator is a coarse way to quantize the joint responses of various filters (one for each neighbor n_i). Likewise, OT-LBP is also a quantizer of these joint responses, but it is built adaptively. Ahonen tested the K-Means algorithm as an adaptive quantizer, but found it to be inferior to LBP for a texture recognition task.

Forests of randomized trees have become a popular option to construct codebooks for computer vision tasks. Moosmann et al [17] use Extremely Randomized Clustering forests to create codebooks of SIFT descriptors [20]). Shotton et al. [16] use random forests to create codebooks for use as features in image segmentation. While the use of trees in these works is similar to ours, they use the results of the quantization in a very different way; the features are given to classifiers such as SVMs, which are not suitable for use in our problem. Furthermore, we have found that for our problem single trees are more effective than random forests.

Wright et al. [21] use unsupervised random forests to quantize SIFT-like descriptors for face recognition. The main difference with our algorithm is that we do not quantize complex descriptors extracted from the image but we work directly on the grayscale image. In addition, the accuracy of their algorithm on the tested datasets is relatively poor compared to other state-of-the-art algorithms. This may be due to the use of an unsupervised algorithm to construct the trees.

There are various recent works using K-Means to construct codebooks to be used for face recognition in a framework similar to ours. Meng et al [22] use it to directly quantize patches from the grayscale image patches. Xie et al [23] as well as [24] use it to quantize patches from images convolved with Gabor wavelets at various scales and orientations. These algorithms are the closest in spirit to our work, since they are partly inspired by LBPs. These algorithms differ from ours in the algorithm used to construct the codebook. They use K-Means, which has the drawback of not being supervised and thus unable to take advantage of labeled data. In addition, for the same number of codes, K-Means are less efficient than trees. Finally, unlike ours, two of the above algorithms incorporate Gabor wavelet features.

Another line of investigation worth mentioning is the use of heuristic algorithms, and in particular evolutionary algorithms, to construct visual descriptors for different purposes. Perez and Olague [25] use Genetic Programming (GP) with a large set of terminals to construct invariant region descriptors for visual matching. Yu and Bhanu [26] also use GP with a large set of operators and Gabor filtering to induce features for facial expression recognition. Kowaliw et al [27] use a variant of GP known as cellular GP to build features for an image classification task. Compared to these approaches, our features simpler, since they do not use a complex set of operations and the tree topology is fixed once the ID3 construction process is finished.

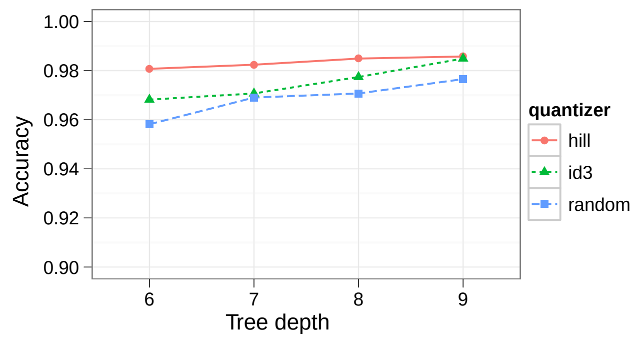
V. Experiments

We perform experiments on the FERET [28] and the CAS-PEAL-R1 [29] benchmark databases. First, we examine the effects of incorporating the hill climbing stage in the tree construction process for each maximum tree depth used. In this case, we measure the accuracy of the algorithm on a subset of FERET. Afterward, we report the accuracy of our algorithm on various standard subsets of FERET and CAS-PEAL-R1 with a selected set of parameters. We report results with and without weights, where the weights for each region are set as the final J value from (2) for the tree of each region.

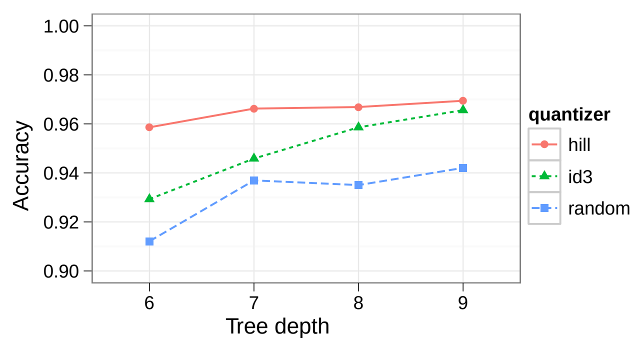
In all images we partition the image into an 7×7 grid, as originally used in [5]. While in general we have found this partition to provide good results, it is likely that adjusting the grid size to each database may yield better results.

For each experiment we show our results along with results from similar works in the recent literature: the original LBP algorithm from Ahonen [5]; the Local Gabor Binary Pattern (LGBP) algorithm, which applies LBP to Gabor-filtered images; the Local Visual Primitive (LVP) algorithm of Meng et al [22], which uses K-Means to quantize grayscale patches; the Local Gabor Textons (LGT) algorithm and the Learned Local Gabor Pattern (LLGP) algorithms, which use K-Means to quantize Gabor filtered-images; and the Histogram of Gabor Phase Patterns (HGPP) algorithms, which quantizes Gabor filtered images into histograms that encode not only the magnitude, but also the phase information from the image. For each algorithm, if a weighting scheme is used, we show the best results with the weighting scheme under the name of the algorithm followed by '-W'.

The results are not strictly comparable, since there may be differences in preprocessing, and for FERET, the training set used, but they provide a meaningful reference.



(a) FERET fb



(b) CAS-PEAL Expression

Figure 4. Accuracy of tree construction schemes and maximum tree depth on the FERET fb and CAS-PEAL Expression datasets. Accuracies for algorithms other than OT-LBP come from the cited papers.

A. *Effect of hill climbing phase and tree depth*

To show the effectiveness of each stage of our tree construction process, we compare the accuracy of a randomly constructed tree (as a baseline), a tree constructed with ID3 and a tree constructed with ID3 and optimized with hill climbing on the FERET *fb* (figure 4a) and CAS-PEAL Expression (figure 4b) datasets.

As expected, the optimized trees perform better than the ID3 trees and the ID3 trees perform better than the random trees. The improvement obtained by hill climbing seems to decrease with larger tree sizes. It is likely this is due of the larger problem space associated to the larger trees, which makes the task for the hill climbing process more difficult.

B. Results on FERET

For FERET, we use *fa* as gallery and *fb*, *fc*, *dup1* and *dup2* as probe sets. For training, we use the FERET standard training set of 762 images from the training CD provided by the CSU Face Identification Evaluation System package [30].

Method	fb	fc	dup1	dup2
LBP [5]	0.93	0.51	0.61	0.50
LBP-W [5]	0.97	0.79	0.66	0.64
LGBP [12]	0.94	0.97	0.68	0.53
LGBP-W [12]	0.98	0.97	0.74	0.71
LVP [23]	0.97	0.70	0.66	0.50
LVP-W [23]	**0.99**	0.80	0.70	0.60
LGT [24]	0.97	0.90	0.71	0.67
HGPP [31]	0.98	**0.99**	0.78	0.76
HGPP-W [31]	0.98	**0.99**	0.78	0.77
LLGP [23]	0.97	0.97	0.75	0.71
LLGP-W [23]	**0.99**	**0.99**	0.80	0.78
OT-LBP$_6^7$	0.98	0.97	0.78	0.74
OT-LBP-W$_6^7$	**0.99**	0.98	0.81	0.78
OT-LBP$_7^7$	0.98	0.97	0.82	0.78
OT-LBP-W$_7^7$	**0.99**	0.98	0.83	0.79
OT-LBP$_8^7$	0.98	**0.99**	0.82	0.78
OT-LBP-W$_8^7$	**0.99**	**0.99**	0.83	0.80
OT-LBP$_9^7$	0.98	**0.99**	0.83	0.79
OT-LBP-W$_9^7$	**0.99**	**0.99**	**0.84**	**0.83**

Table I
ACCURACY ON FERET PROBE SETS. OT-LBP$_d^r$ CORRESPONDS TO A TREE OF MAXIMUM DEPTH d AND RADIUS r. OT-LBP-W IS THE WEIGHTED VERSION OF OT-LBP. ACCURACIES FOR ALGORITHMS OTHER THAN OT-LBP COME FROM THE CITED PAPERS.

We can see that our algorithm does well on FERET. It obtains the best results on all the datasets.

C. Results on CAS-PEAL-R1

In CAS-PEAL-R1 we use the standard training and gallery subsets, and we use the Expression, Lighting and Accessory subsets as probes.

In this dataset our algorithm performs comparably with other algorithms in the Expression and Accessory datasets. On the lighting dataset, the overall performance of all the algorithms is rather poor. In this case, the best performance are given by LGBP, HGPP and LLGP. All these algorithms use features based on Gabor wavelets, which suggests that Gabor features provide robustness against the extreme lighting variations in this dataset.

Method	Expression	Accessory	Lighting
LGBP [12]	0.95	0.87	0.51
LVP [22]	0.96	0.86	0.29
LVP-W [22]	0.96	0.86	0.33
HGPP [31]	0.96	**0.92**	**0.62**
HGPP-W [31]	0.97	**0.92**	**0.63**
LLGP [23]	0.96	0.90	0.52
LLGP-W [23]	0.96	0.92	0.55
OT-LBP$_6^7$	0.96	0.89	0.33
OT-LBP-W$_6^7$	0.97	0.89	0.34
OT-LBP$_7^7$	0.97	0.90	0.36
OT-LBP-W$_7^7$	0.97	0.90	0.36
OT-LBP$_8^7$	0.97	0.90	0.37
OT-LBP-W$_8^7$	0.97	0.90	0.37
OT-LBP$_9^7$	0.97	0.90	0.37
OT-LBP-W$_9^7$	**0.98**	0.91	0.37

Table II
ACCURACY ON CAS-PEAL-R1 PROBE SETS. OT-LBP$_d^r$ CORRESPONDS TO A TREE OF MAXIMUM DEPTH d AND RADIUS r. OT-LBP-W IS THE WEIGHTED VERSION OF OT-LBP.

VI. CONCLUSIONS AND FUTURE WORK

We have proposed a novel method that uses training data to create discriminative LBP-like descriptors by using decision trees. The algorithm obtains encouraging results on standard databases, and presents better results that several state-of-the-art alternative solutions. In particular, with respect to a face recognizer based on the widely used LBPs, our approach presents a significant increase in accuracy, demonstrating the advantages of using an adaptive and discriminative set of local binary patterns.

Regarding future work, seeing the good performance of algorithms that use features based on Gabor wavelets (such as [23] and [31]) we are incorporating these type of features into our algorithm.

ACKNOWLEDGEMENTS

This work was partially funded by FONDECYT grant 1095140 and LACCIR Virtual Institute grant No. R1208LAC005 (http://www.laccir.org).

Portions of the research in this paper use the FERET database of facial images collected under the FERET program.

The research in this paper uses the CAS-PEAL-R1 face database collected under the sponsorship of the Chinese National Hi-Tech Program and ISVISION Tech. Co. Ltd.

REFERENCES

[1] J. Zou, Q. Ji, and G. Nagy, "A comparative study of local matching approach for face recognition," *IEEE Trans. Image Process.*, vol. 16, no. 10, pp. 2617–2628, 2007.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[3] M. Bicego, A. Lagorio, E. Grosso, and M. Tistarelli, "On the use of SIFT features for face authentication," in *CVPR*, 2006, p. 35.

[4] A. Albiol, D. Monzo, A. Martin, J. Sastre, and A. Albiol, "Face recognition using HOG-EBGM," *Pattern Recognition Letters*, vol. 29, no. 10, pp. 1537–1543, 2008.

[5] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2037–2041, 2006.

[6] J. Ruiz-del-Solar, R. Verschae, and M. Correa, "Recognition of faces in unconstrained environments: A comparative study," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, pp. 1–20, 2009.

[7] T. Ojala, M. Pietikinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[8] X. Tan and B. Triggs, "Enhanced local texture feature sets for face recognition under difficult lighting conditions," in *Proc. of the 3rd International Conference on Analysis and Modeling of Faces and Gestures*, 2007, pp. 168–182.

[9] S. Liao and A. C. S. Chung, "Face recognition by using elongated local binary patterns with average maximum distance gradient magnitude," in *ACCV*, Berlin, Heidelberg, 2007, pp. 672–679.

[10] S. Liao, X. Zhu, Z. Lei, L. Zhang, and S. Li, "Learning multi-scale block local binary patterns for face recognition," in *Advances in Biometrics*, 2007, pp. 828–837.

[11] L. Wolf, T. Hassner, and Y. Taigman, "Descriptor based methods in the wild," in *Real-Life Images Workshop at ECCV*, October 2008.

[12] W. Zhang, S. Shan, W. Gao, X. Chen, and H. Zhang, "Local gabor binary pattern histogram sequence (LGBPHS): A novel non-statistical model for face representation and recognition," in *ICCV*, 2005.

[13] D. Maturana, D. Mery, and S. Álvaro, "Face recognition with decision tree-based local binary patterns," in *ACCV*, 2010.

[14] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[15] J. Koza, *Genetic programming: on the programming of computers by means of natural selection*. The MIT press, 1992.

[16] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *CVPR*, 2008.

[17] F. Moosmann, E. Nowak, and F. Jurie, "Randomized clustering forests for image classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 9, pp. 1632–1646, 2008.

[18] A. Bosch, A. Zisserman, and X. Muñoz, "Image classification using random forests and ferns," in *ICCV*, 2007.

[19] T. Ahonen and M. Pietikinen, "Image description using joint distribution of filter bank responses," *Pattern Recognition Letters*, vol. 30, no. 4, pp. 368 – 376, 2009.

[20] D. G. Lowe, "Distinctive image features from Scale-Invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[21] J. Wright and G. Hua, "Implicit elastic matching with random projections for pose-variant face recognition," in *CVPR*, 2009, pp. 1502–1509.

[22] X. Meng, S. Shan, X. Chen, and W. Gao, "Local visual primitives (LVP) for face modelling and recognition," in *ICPR*, 2006.

[23] S. Xie, S. Shan, X. Chen, X. Meng, and W. Gao, "Learned local gabor patterns for face representation and recognition," *Signal Processing*, vol. 89, no. 12, pp. 2333 – 2344, 2009, special Section: Visual Information Analysis for Security.

[24] Z. Lei, S. Li, R. Chu, and X. Zhu, "Face recognition with local Gabor textons," *Advances in Biometrics*, pp. 49–57, 2007.

[25] C. B. Perez and G. Olague, "Learning invariant region descriptor operators with genetic programming and the f-measure," in *ICPR*, 2008, pp. 1–4.

[26] J. Yu and B. Bhanu, "Evolutionary feature synthesis for facial expression recognition," *Pattern Recogn. Lett.*, vol. 27, no. 11, pp. 1289–1298, 2006.

[27] T. Kowaliw, W. Banzhaf, N. Kharma, and S. Harding, "Evolving novel image features using genetic programming-based image transforms," in *CEC*, 2009, pp. 2502–2507.

[28] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The FERET evaluation methodology for Face-Recognition algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 10, pp. 1090–1104, 2000.

[29] W. Gao, B. Cao, S. Shan, X. Chen, D. Zhou, X. Zhang, and D. Zhao, "The CAS-PEAL large-scale Chinese face database and baseline evaluations," *IEEE Trans. Syst., Man, Cybern. A*, vol. 38, no. 1, pp. 149–161, 2008.

[30] D. Bolme, J. Beveridge, M. Teixeira, and B. Draper, "The CSU face identification evaluation system: Its purpose, features and structure," in *ICCV*, 2003.

[31] B. Zhang, S. Shan, X. Chen, and W. Gao, "Histogram of gabor phase patterns (HGPP): A novel object representation approach for face recognition," *IEEE Trans. Image Process.*, vol. 16, no. 1, pp. 57–68, 2007.