Visual Recognition using Local Quantized Patterns

Sibt ul Hussain and Bill Triggs
sibt.ul.hussain@gmail.com, bill.triggs@imag.fr

Laboratoire Jean Kuntzmann
BP 53, 38041 Grenoble, France.

Abstract. Features such as Local Binary Patterns (LBP) and Local Ternary Patterns (LTP) have been very successful in a number of areas including texture analysis, face recognition and object detection. They are based on the idea that small patterns of qualitative local gray-level differences contain a great deal of information about higher-level image content. Existing local pattern features use hand-specified codings, which limits them to small spatial supports and coarse graylevel comparisons. We introduce Local Quantized Patterns (LQP), a generalization that uses lookup-table based vector quantization to code larger or deeper patterns. LQP inherits some of the flexibility and power of visual word representations, without sacrificing the run-time speed and simplicity of existing local pattern ones. We show that it outperforms well-established features including HOG, LBP and LTP and their combinations on a range of challenging object detection and texture classification problems.

1 Introduction

"Local pattern" features such as Local Binary Patterns (LBP) [1,2,3], Local Ternary Patterns (LTP) [4]) and Weber Law Descriptors (WLD) [5] are based on the idea that small patterns of qualitative local gray-level differences contain a great deal of information about higher-level image content. Despite their extreme simplicity and micro-locality, local patterns have proven very successful in visual recognition tasks ranging from texture classification to face analysis and object detection [3]. For example, [6] shows that LTP outperforms established feature sets such as HOG on the PASCAL VOC'06 object detection challenge, and that combining HOG, LTP and LBP gives even better results.

In many applications, local information is harnessed by finding a "local pattern coding" of the image – a qualitative local gray-value pattern associated with each image pixel – and counting the number of occurrences of each possible pattern over suitable image regions – *e.g.* a grid of rectangular cells overlying the putative object. The resulting histograms are used as visual descriptors for recognition. This approach works surprisingly well in practice [3], however current local pattern descriptors use hard-wired codings that severely limit the pattern shapes and sizes that can used, and they rely on very coarse qualitative gray-value comparisons, typically binarization. It is worth asking whether variants incorporating different shapes such as filled rectangles or strips of pixels, more pixels, or finer comparisons would be even more effective.

The problem is that the code size (the number of different local patterns that can occur), and hence the size of the cell-level histograms, increases exponentially with the size of the spatial support of the pattern and the number of quantization levels. Specifically, quantizing a group of n pixel gray-level comparisons into k levels gives a code of size k^n. For practical object recognition tasks we would like to limit the histograms to at most a few hundred entries per cell, which limits the size of the local patterns to at most about 8 binary pixel comparisons or 5 ternary ones. As a result, in such applications both LBP and LTP typically sample 8 pixels on a one pixel radius circle around the central pixel. LBP [3] then thresholds these at the gray-value of the central pixel to produce a $2^8 = 256$ valued coding, whereas LTP [4] introduces a parameter τ, thresholds at $-\tau$ and τ to generate 3-way comparisons, then splits the resulting ternary codes into "positive" and "negative" binary halves to produce two 256-valued codings. The splitting is necessary because a $3^8 = 6561$ valued code would not be practicable. Although it is rather heuristic, it seems to work well in practice [4,6].

Furthermore, in both cases one typically postprocesses the output to produce "uniform" codings. Ojala *et al* [1] observed that of the 256 possible code values, the "uniform patterns" – ones that have at most one island of ones and one island of zeros around the sampled circle – both occur most frequently and carry most of the discriminative power. There are 58 of these, and the counts for the remaining patterns are typically pooled into a single additional "non-uniform" bin, giving 59 bin output histograms.

Although the splitting and uniform coding dimensionality reductions are well-established, it should not be forgotten that they are at best limited palliatives for a more serious underlying problem – the exponential growth of codebook size with neighbourhood size and quantization depth. It is unclear how much information is lost during splitting and whether taking positive and negative halves is the best way to organize the split. Similarly, it is unclear how best to generalize uniform coding to non-circular spatial topologies and to ternary or higher-order codes, and even for binary circular ones, the quadratic increase in the number of uniform patterns with circumference limits uniform coding to groups of at most about 20 pixels (and even then, the quality of the coding is likely to suffer as the fraction of the 2^n codes that are uniform rapidly becomes infinitesimal). Nor is it obvious that assigning all of the nonuniform codes to a single histogram bin is the best solution – it might be better to assign each nonuniform code to the "nearest" (in some sense) uniform one. In general it seems likely that the need for such hand-coded reductions is limiting the size and expressiveness of local pattern representations, so it is worthwhile to seek methods for learning efficient reductions from very large input codings ($e.g.$ k^n) to much smaller output codings that can be stored compactly.

The same problem occurs in the popular "visual words" approaches [7,8,9], where (potentially) continuous-valued feature vectors for larger image patches are reduced to a limited set of code values by vector quantization methods such as K-Means. These methods can also be applied to local pattern representations, but even though they would resolve many of the issues mentioned above, they are usually too slow to be practicable. The problem is that they typically require each incoming sample to be compared to every dictionary element to discover its quantization class, or at very least they require an expensive data structure traversal for each sample at run time. This is already problematic for patch-level descriptors in visual word representations, so it is much too slow for practical object detectors that need to evaluate the local pattern of every pixel in the image pyramid. To be useable, the cost of run-time code evaluation needs to be limited to a constant-time operation per pixel – *e.g.* a closed form formula or a table lookup[1].

Our Local Quantized Pattern (LQP) approach does just this. We use vector quantization for codebook learning, but choose pattern sizes that allow run-time local pattern coding to be implemented by simple table lookup, thus retaining many of the advantages of both current local patterns and visual words in an efficient and practical form that remains well-suited to hardware implementation.

There are other methods that harness quantized local neighbourhood information for visual recognition tasks, and these too suffer from the compromise between code size and quantization time / code quality that LQP is designed to address. For example, the face recognition method of Cao *et al* [11] takes patches from nine fiducial regions on pre-detected faces and in each samples 24 preprocessed grayvalues on a double ring around each pixel (the same sampling pattern as our Disk$_5$ descriptors below), converting these to very high (up to 2^{17}) dimensional codes using a random projection tree coder followed by PCA to reduce the output dimensionality. Such a costly projection process is acceptable in their context where descriptors are computed in only a few known patches, but it would be impractical for scanning window object detection where dense multiscale evaluation over the whole image is required. Similarly, Calonder *et al* [12] use random pairwise graylevel pixel or smoothed pixel comparisons to produce 128–512 bit binary patch descriptors for keypoint matching. This happens at a patch level and it converts input graylevels to long binary codes (ones much too long to histogram – 2^{128} entry histograms are not practicable), whereas LQP happens at a pixel level and converts moderate-length binary or ternary codes to shorter vector quantized ones.

The basic LQP approach is sketched in the next section. The subsequent sections give further details and experimental results.

2 Local Quantized Patterns

The essence of LQP is to apply visual word quantization to discrete local patterns, using a precompiled lookup table to cache the final coding for speed. The main constraint is the size of the lookup table, which must store the output code for every possible input pattern. For example, for a local pattern that includes

[1] See [10] for a visual word approach with similar motivations.

24 binary pixel comparisons the table has $2^{24} = 17$ million entries, while for one including 16 ternary pixel comparisons it has $3^{16} = 43$ million entries and for one including 10 5-level pixel comparisons it has $5^{10} = 9.8$ million. These are reasonable upper limits for implementation in hardware, so in practice LQP allows patterns that are around three times larger than existing local pattern approaches – going from 8 to 24 pixels for binary coding and from 5 to 16 for direct ternary coding. Alternative data structures may allow these sizes to be increased somewhat, but for simplicity we will only test direct table lookup here. Note that as typical codebooks provide only $O(10^2)$–$O(10^3)$ output codes, $O(10^5)$ input patterns map to each output code.

We will see that by allowing larger local pattern neighbourhoods to be used, LQP provides a significant increase in discriminative power. It also allows a much wider range of neighbourhood shapes, and it adapts the coding to the application, the neighbourhood shape, and the desired output size. Moreover – although we will only test simple L_2 pattern comparison here – LQP's table-lookup structure allows it to handle more sophisticated pattern comparison metrics such as Earth Mover style distances and coding under symmetries such as rotations and reflections [3], all at no additional run-time cost.

The lookup table architecture can also be used to accelerate codebook learning. During learning the (codebook) training dataset is scanned once to record the number of occurrences of each input code that occurs, storing these in a hash table or index, then these values and counts are passed to the algorithm that actually learns the codebook, *e.g.* a count-weighted version of K-Means. This typically makes training fast because most of the possible input codes do not actually occur and for those that do, all occurrences are processed in a single operation. *E.g.* for 24 bit binary vectors over the INRIA Person positive training set, only 671k of the 17M possible input values actually occur and as a result 10 rounds of K-Means for a 100 element dictionary takes only 12 minutes.

2.1 Local Pattern Geometry

As mentioned above, within the above size limits (24 pixel-comparisons for binary and split ternary coding, 16 for full ternary coding), many different neighbourhood geometries are possible for LQP and one of its main advantages is its flexibility in this respect. Here we test a selection of possible geometries including: horizontal (H), vertical (V), diagonal (D) and antidiagonal (A) strips of pixels; combinations of these like horizontal-vertical (HV), diagonal-antidiagonal (DA) and horizontal-vertical-diagonal-antidiagonal (HVDA); and traditional circular and disk-shaped regions – *c.f.* Figure 1. By default we compare each non-central pixel to the central one, but we also tested "center symmetric" (CS) codes, where each pixel is compared to the diametrically opposite one. Geometries will be described by notation such as HV_7^3, where HV denotes the neighbourhood shape (here a horizontal-vertical cross), the subscript indicates the neighbourhood diameter (here 7 pixels) and the superscript indicates the quantization level (here native ternary coding – 3* denotes split ternary coding and 2 binary coding). Separate codebooks are learned for the positive and negative

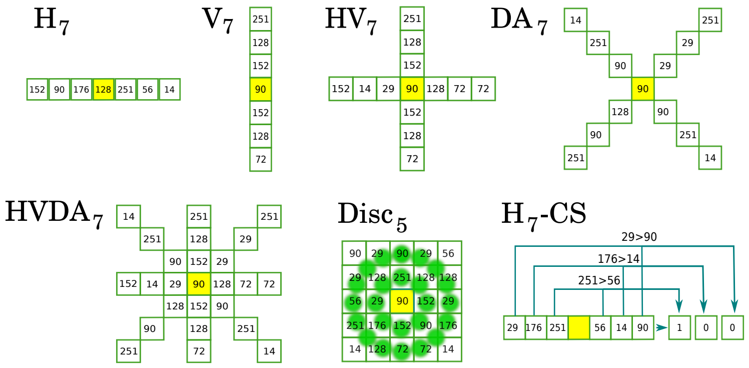

Fig. 1. Some examples of the local pattern neighbourhoods that we have tested. The subscript denotes the diameter in pixels of the local pattern neighbourhood. A superscript will be added to denote the depth of quantization: 2 for binary coding, 3 for full ternary coding, and 3* for split ternary coding. In each case, pixels are sampled around a central pixel (shaded yellow) and compared either with the central pixel or with the diametrically opposite pixel (CS case).

halves of split ternary codes. The results below show that when using ternary patterns, it is almost always preferable to split the code to allow a larger spatial support to be used.

As always, codes from different feature sets or from different types of LQP neighbourhoods can be combined at the cell level by simple histogram concatenation. This effectively allows more pixels to be incorporated in the local pattern neighbourhood, at the cost of ignoring co-occurrences between the different sub-patterns. However it should be noted that elongated LQP's such as the strips H_{25} and V_{25} are often so wide that they cross several cells, making them difficult to handle owing to boundary effects.

2.2 Codebook Learning

By default we represent input patterns as binary or ternary vectors, using K-Means [13] over the standard L_2 inter-vector distance to learn our visual codebooks. Our algorithm counts the number of occurrences of each possible input code on the training set, using the resulting sparse table to run K-Means. It is run ten times with different random initializations, taking the codebook from the run that gives the lowest K-Means quantization error on the training set. Below we use conventional hard K-Means coding. We also tested a soft-coding variant in which the lookup table stored the $c \leq 10$ nearest code centers for each input and the incoming patterns were soft-quantized against all c codes using uniform weighting, but we found that this reduced the accuracy.

For comparison, we also implemented a fast discriminative coding method. We trained Random Forests of classification trees over raw (unquantized) pixel differences, using them to construct tree-leaf-level output codes in the fashion

	LBP	LTP	Disk_5^{3*}	CWLD	WLD	WLD	MWLD
Brodatz	90.6	95.7	96.9	89.3	89.3	97.1	-
KTH	58.7	60.7	64.2	56.5	59.4	56.4	64.7
CUReT	88.1	88.7	92.2	83.8	82.2	-	-

Table 1. Texture classification accuracies for various features and datasets. Columns 4-5 show results for our implementations of color and grayscale WLD, while columns 6-7 reproduce grayscale WLD and MWLD results from [5].

of [14,15]. To capture the diversity of the datasets, relatively large numbers of relatively shallow trees were used. Although Random Forests do not require a preliminary quantization of their input values, we found that their output codings were less accurate than K-Means based ones, as well as being much slower to train and use. This is perhaps because distance-based methods like K-Means provide relatively smooth decompositions of the input space, whereas – notwithstanding their discriminative nature and unquantized inputs – Random Forests only provide rather coarse random partitionings. For instance (anticipating the results below), for the H_7^3+V_7^3 feature combination with single root Latent SVM detectors on the INRIA Person dataset, the K-Means coding with 100 centers and hence 200 histogram bins gave an Average Precision of 65.3%, whereas the Random Forest ones with 16 trees of respectively 8 leaf nodes (128 bin histograms) and 16 leaf nodes (256 bin histograms) gave only 57.7% and 59.0%. For this reason, we give results only for K-Means in the experiments below.

3 Texture Classification Experiments

We will provide a detailed experimental study of LQP for object detection below, but first we anticipate the settings found there to give some brief results on the simpler problem of texture classification. We test LQP on the Brodatz32 [16,17], CUReT [18][2] and KTH-TIPS-2a [19] datasets. In each case the descriptor is the LQP histogram of the entire unpreprocessed input image. To simplify comparison to previous work we present accuracies for simple 3-nearest-neighbour classifiers: better results are known to be available with more sophisticated classifiers such as SVM's [19]. We tested both the χ^2 and Normalized Histogram Intersection metrics for histogram comparison, reporting for χ^2 as it gave better results. We tested various normalization schemes for the LQP histograms. Unnormalized histograms and L2 normalization gave the equal-best results, so we report results for unnormalized histograms.

We follow the protocols of [5,19,9]. For Brodatz [5], the data for each class is randomly split into equal halves with half used for training and the other half for testing, and we report averages over 10 random trials. For KTH [19], three samples of each class are used for training and the fourth for testing, and we report averages over four random partitions. For CUReT [9], we split the images

[2] From http://www.robots.ox.ac.uk/∼vgg/research/texclass/index.html

of each class into equal parts and use half for training and the remaining half for testing.

Table 1 summarizes the results for the Disk$_5^{3*}$ LQP features. In all three cases, LQP outperforms LTP, which in turn outperforms LBP. We also tested against WLD and multi-scale WLD (MWLD), which concatenates WLD histograms from several different spatial supports and scales [5]. We were not able to reproduce the results of [5] with our grayscale WLD implementation[3], doing worse on Brodatz and better on KTH, so we give figures for both implementations and also for our color WLD to allow comparison with the other features shown. Overall, LQP appears to outperform WLD. A fair comparison with MWLD is more difficult as results for it [5] are available only on KTH and its feature dimension is about 6 times higher than that of LQP.

4 Object Detection Experiments

For object detection, we give results on the INRIA Person [20] and PASCAL VOC2006 [21] datasets, using the INRIA set mainly for parameter setting. We follow VOC2006 protocols for both datasets, reporting Average Precisions (AP's), *i.e.* areas under Precision-Recall curves. Our detectors are based on those of [20,22,6]. For each feature set tested, we overlay the image with a grid of 8×8 pixel cells, using bilinear spatial interpolation to accumulate votes into the cell-level histograms in order to provide robustness to small spatial displacements. The histograms are gathered into detector-window level feature vectors and used in Latent SVM sliding window detectors in the manner of [22]. Our main aim here is to evaluate LQP on a broad range of problems, not to get the absolute best possible results on each individual one, so we only report results for detectors with a single bilaterally-symmetric root and no parts, trained using the densified version of SVMLight [23,20]. Past experience suggests that including multiple roots and parts will give better results for all of the features tested without significantly changing their relative rankings.

For HOG we use the HOG$_{31}$ features of [22] by default – these include both signed and unsigned gradient orientations and dimensionality reduction over the different normalizations of a cell. For LBP and LTP we follow [6]. Circular 8-sample patterns of radius 1 pixel are used, with bilinear interpolation to resample the pixels needed from the unpreprocessed input image, and uniform coding is used, with splitting for LTP. For color images, patterns are evaluated separately on the R, G and B color channels then accumulated into a shared histogram. L_1-Sqrt normalization is used for histograms, *i.e.* they are normalized to sum 1 then square-rooted. Our LQP features use exactly the same approach, simply substituting the LQP code for the LBP/LTP one, without uniform coding but with splitting as necessary.

[3] We do not apply the empirical histogram reweighting from [5] as we find that it makes the results much worse, but in either case our results differ from [5].

Diameter	H^3	V^3	D^3	H^2	V^2	D^2
7	63.3	61.4	60.5	-	-	-
9	64.9	64.8	61.8	46.4	49.0	47.1
15	67.2	68.4	62.3	54.7	53.8	51.4

Table 2. AP's for various detectors trained using single-strip LQP features on the INRIA Person dataset. The missing values correspond to features that have so few input codes that they can be coded directly without using LQP, *e.g.* H^2_7 has $2^7 = 128$ codes in total.

Codebook Size	50	60	80	100	150	200	300
HV^3_5	76.8	76.4	76.1	77.9	77.0	80.0	79.9
$HVDA^3_5$	77.3	77.0	79.8	78.7	79.9	81.5	82.3
HV^3_7	77.6	77.1	77.8	79.5	79.1	79.6	81.4
DA^3_7	75.1	76.1	76.0	77.0	76.3	78.4	79.1
$HVDA^{3*}_7$	80.1	81.1	80.2	80.9	81.7	82.0	82.6
$Disc^{3*}_5$	79.3	79.9	81.3	81.2	82.2	81.3	81.4

Table 3. The effect of different LQP geometries and codebook sizes on the AP's of single root latent detectors on the INRIA Person dataset. Note that for split ('3*') features, the final descriptor size is twice the quoted codebook size.

4.1 INRIA Person Dataset

Strip Layouts. Table 2 shows AP's on the INRIA Person dataset for detectors using LQP features based on single horizontal, vertical or diagonal strips of pixels. The single-strip features turn out to be quite weak, giving significantly lower performance than existing feature sets – LBP, LTP and HOG give respectively 74.0%, 79.0% and 79.0% AP on this dataset. However it is at least clear that increasing the length of the strip increases the performance, as does replacing binary codes with ternary ones. The comparable centre-symmetric features (which compare each pixel with its diametrically opposite one, not with the centre one, thus halving the size of the input pattern) consistently give much lower performance – *e.g.* H^3-CS and V^3-CS respectively have AP's of 59.3% and 59.1% – so we will not test them further.

Cross Layouts. Including several complementary strips in the LQP neighbourhood significantly increases the accuracy – *c.f.* Table 3. For instance, for a 100 word dictionary the cross layout HV^3_7 gives 79.5% AP whereas the cell-level concatenation of the 100 word H^3_7 and V^3_7 LQP histograms gives only 74.6%. If split uniform LTP coding is used instead of LQP coding for the H^3_7 and V^3_7 histograms, the results are still worse – 60.4% AP for a $4 \times 33 = 132$ dimensional histogram. Clearly, the richer co-occurrence statistics that HV^3_7 LQP captures are more useful than the 100 extra codewords of H^3_7+V^3_7. Despite the inclusion of only two orientations, the HV^3_7 results are already slightly better than HOG and LTP on this dataset, both of which give 79.0% AP. In contrast, DA^3_7, which combines diagonal and antidiagonal strips, gives only 77.0% AP– presumably horizontal and vertical slices are more discriminant for people than diagonal ones. Incorporating all four types of strip in a "Union Jack" pattern HVDA further improves the results, although split coding must be used in this case owing to the number of pixels in the pattern.

Disk Layouts. Disk-shaped patterns can do even better. Using 100 word codebooks on the INRIA dataset, the two-ring 24 pixel pattern $Disk^{3*}_5$ (81.2% AP) outperforms both the 16 pixel ($HVDA^3_5$, 78.7% AP) and 24 pixel ($HVDA^{3*}_7$, 80.9% AP) Union Jacks, and also LTP and HOG (both have 79.0% AP) – *c.f.* Table 3. The VOC2006 results below confirm that Disk$_5^{3*}$ has slightly better overall performance than HVDA$_7^{3*}$: presumably, dense circular sampling in a compact neighbourhood captures more of the characteristic class structure than sampling a fixed set of rays in the broader neighbourhood covered by HVDA$_7^{3*}$. In fact, Disk$_5^{3*}$ gives better results on the INRIA Person dataset than any previous individual or combined feature set that we are aware of, improving the AP by 2.2% (3.2% for the 150 word codebook) relative to the individual feature baselines HOG and LTP and by 0.2% (for the 150 word codebook) relative to the combined feature baseline LBP+LTP+HOG [6].

Haar Layout. To see whether it would be useful to include multiscale information (*c.f.* MWLD [5]), we also tested a Haar wavelet based local pattern. We take 4×4 pixel neighbourhoods around each pixel, apply the Haar wavelet transform, discard the constant term and code the remaining 15 wavelet coefficients using LQP ternary coding. In detail, this involves taking the four 2×2-pixel corner blocks of the neighbourhood and a 2×2 block containing the average of each corner one, and applying 2×2 horizontal, vertical and diagonal Haar filters [24,25] to each block. The resulting Haar LQP features give slightly better results than HOG and LTP: for 100 word codebooks, 80.7% AP versus 79% AP on INRIA Person, and 29.4% AP versus 25.1% and 28.9% AP on the VOC2006 person class. However the Haar patterns do not equal the performance of the best Disk and HVDA ones on these datasets.

Splitting and LQP Features. With a 118 word codebook, Disk$_3^3$ (the LQP form of LTP's 8-sample circle of 1 pixel radius) gives identical accuracy to traditional 118-D split uniform LTP coding. Reducing the LQP codebook size to 88 reduces the accuracy by only 0.8%. For the 16-sample radius 2 circle Circ$_5^3$ (the largest pattern for which unsplit ternary coding is feasible), split coding with 100 word codebooks (200-D histograms) gives 80.9% AP on INRIA whereas unsplit coding with codebook (and histogram) sizes of 100 and 200 gives respectively 78.8% and 80.5% AP. Similarly, for the VOC2006 person class, split Circ$_5^{3*}$ gives 28.3% AP while unsplit Circ$_5^3$ gives respectively 26.0% and 28.6% AP for 100 and 200 word codebooks, and for the VOC2006 car class split Circ$_5^{3*}$ gives 55.0% AP while unsplit Circ$_5^3$ gives 55.2 % and 54.9% AP. These results in some sense validate the use of split uniform coding in the original LTP. Overall, our results consistently show that splitting causes little loss of discriminative power relative to the equivalent unsplit coding, and that it is beneficial in the sense that it allows larger spatial supports to be used, thus increasing the overall discriminative power.

Ternary Code Threshold. Figure 2 shows the effect of the ternary code threshold τ on the APs of various LQP features on the INRIA Person and VOC2006 person classes. For each feature there is a broad range of τ values that gives similar results, but spatially larger patterns (notably the extended cross layouts) need larger values of τ, presumably because the typical ranges of gray-value variations increase with increasing pattern diameter. Over the full set of classes,

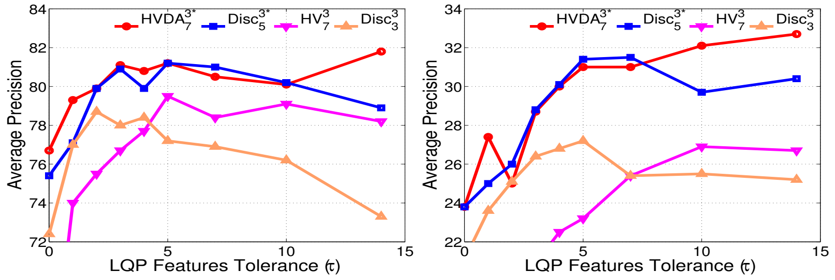

Fig. 2. Average Precisions of LQP features on the INRIA Person (left) and VOC2006 person (right) datasets, for different values of the ternary quantization threshold τ.

$\tau = 5$ turns out to be the best value for both of the Disk diameters and also for LTP, whereas $\tau = 14$ is the preferred value for broad crosses such as HVDA$^{3*}_7$.

Choice of Codebook. Table 3 shows how codebook size affects the performance of various LQP features. As expected – despite some variability owing to the nonconvexity of K-Means learning – larger codebooks typically have better performance. By default we use 100 word codebooks below as they seem to offer a reasonable compromise between descriptor size and performance, but smaller ones still offer very respectable levels of performance and larger ones are often even better.

More generally, features can be quantized using either a single codebook (learned from the positives, the negatives, or the complete training set), or several concatenated codebooks – for example ones learned separately on the positive and negative training sets. Moreover, for positive training we can also learn a separate "cell level" codebook for each cell of the detection window, subsequently quantizing the pixels of each cell of the current window using that cell's positive codebook and the global negative one.

Table 4 shows the effect of these different codebook learning schemes on the accuracy of INRIA Person detectors using HV$^3_7$ features. The single-codebook results are better than the multiple-codebook ones and in the multiple codebook case, learning separate cell-level positive codebooks provides only a small increase in the AP so it does not seem to be warranted given its extra complexity. Unsurprisingly, using positives alone for codebook learning is better than using negatives alone – the positive codebooks are trained on structures that are important for characterizing the object class – but (perhaps surprisingly) pooling the positives and negatives during training gives worse results than using either positives or negatives alone. For the binary and split ternary codings, initializing some of the K-Means centers at the LBP/LTP uniform patterns to encourage the latter to be well coded does not change the performance. Similarly, for global positive and negative codebooks, learning the negative codebook first and using it to initialize the positive one does not change the performance. By default, we therefore use single 100 word codebooks obtained by running K-Means on (all of the R, G and B pixels of the annotation windows of) the positive training data.

Codebook Type	Positive	Negative	Combined	Positive & Negative	Cell Based
Cell Dimension	100	100	100	200	200
HV_7^3	79.5	78.8	77.8	77.0	78.0

Table 4. The influence of different codebook organizations on the AP's of HV_7^3 detectors on the INRIA Person dataset.

	HV_7^3	DA_7^3	$HVDA_5^3$	$HVDA_7^{3*}$	$Disk_5^{3*}$	$HVDA_9^3$-CS
LQP	79.5	77.0	78.7	80.9	81.2	78.5
LQP+HOG	81.7	80.5	82.8	82.7	**82.8**	81.8

Table 5. Average Precisions on the INRIA Person dataset for detectors using classical HOG [20] plus the given ternary LQP feature with a 100 word codebook. For comparison, HOG alone gives 79.0%, LTP+HOG gives 81.3% and LBP+LTP+HOG gives 82.0% AP [6].

LQP Type	CB Size	Desc Size	**Mean**	Bike	Bus	Car	Cat	Cow	Dog	Horse	Mbike	Person	Sheep
HV_7^3	100	100	34.6	55.8	39.8	55.1	16.0	32.8	14.8	21.8	46.9	25.0	38.2
	150	150	35.6	56.4	42.0	54.5	16.1	36.5	15.8	22.2	47.6	25.8	38.9
$HVDA_5^3$	300	300	36.2	52.8	43.5	**56.8**	14.9	35.9	17.3	26.7	48.4	26.8	38.7
$HVDA_7^{3*}$	100	200	38.3	**58.2**	45.9	56.1	14.6	38.0	17.4	27.7	52.4	32.7	39.8
	150	300	39.4	58.0	46.5	56.5	19.0	**39.0**	17.6	28.9	**52.7**	**33.0**	42.6
$Disk_5^{3*}$	100	200	38.8	57.8	46.2	56.1	19.3	37.6	18.8	30.9	50.2	29.9	41.2
	150	300	**39.8**	57.9	**49.0**	56.3	20.1	37.7	**18.9**	**32.1**	50.5	31.9	**43.1**
HOG	31	31	31.6	57.2	40.1	55.0	5.0	31.9	5.3	22.0	41.7	25.1	32.2
LBP	59	59	33.2	54.0	39.5	53.8	15.9	33.0	9.2	21.1	44.8	21.8	38.4
LTP	59	118	37.8	56.2	45.2	56.1	17.3	35.8	16.8	29.8	51.4	28.9	40.0
LBP+LTP+HOG	149	208	38.3	57.9	44.4	56.0	18.7	37.0	16.0	29.0	51.2	32.8	39.8

Table 6. Average Precisions of single root latent detectors on the VOC2006 test set using HOG_{31}, LBP, LTP and LQP features. The results for LBP+LTP+HOG are from [6]. $Disk_5^{3*}$ outperforms all three of HOG, LBP and LTP, and even their combination LBP+LTP+HOG.

Combination with HOG Features. On the INRIA Person dataset, the LQP features are so strong that the (single root latent) classifiers used are nearly saturated, so – particularly for strong performers such as $Disk_5^{3*}$ – including additional features such as HOG provides only modest gains in accuracy, *c.f.* Table 5. However the results for VOC2006 below show that (like other local patterns) LQP features complement HOG well on more challenging datasets.

4.2 PASCAL VOC2006

LQP features also give state of the art results on the VOC2006 dataset – *c.f.* Table 6. For instance, HV_7^3 outperforms HOG_{31} on 8 of the 10 classes, increasing the Mean AP by 4%, and LBP on all 10, increasing the Mean AP by 2.4%. LTP outperforms HV_7^3 on 8 of the 10 classes, but $Disk_5^{3*}$ outperforms LTP on 9 of the 10, increasing the Mean AP by 1.0% (and by 2.0% for 150 word codebooks).

	HV_7^3	DA_7^3	$HVDA_5^3$	$HVDA_7^{3*}$	$Disk_5^{3*}$	$HVDA_9^3$-CS
LQP	25.0	20.7	26.1	30.2	29.9	17.4
LQP+HOG	33.8	30.1	33.1	33.6	**34.8**	28.8

Table 7. Average Precisions on the VOC2006 person class for detectors using classical HOG ([20]) plus the given LQP feature. The local patterns use 100 word codebooks. For HOG alone, the AP is 24.1%, while for LTP+HOG it is 33.8%.

$Disk_5^{3*}$ also outperforms $HVDA_7^{3*}$ on 5 of the 10 classes, increasing the Mean AP by 0.4%, and the combination LBP+LTP+HOG [6] on 7 of the 10, increasing the Mean AP by 0.5% despite being lower dimensional. Similarly, for 150 word codebooks $Disk_5^{3*}$ outperforms LBP+LTP+HOG on 7.5 of the 10 classes, increasing the Mean AP by 1.5%. These improvements occur for both structure-dominated classes such as cars and people, and texture-dominated ones such as cats and dogs, so LQP seems to be able to capture both types of cues. To the best of our knowledge, LTP and LBP+LTP+HOG were respectively the individual and combined feature sets with the best reported performance on both INRIA Person and VOC2006 (at least for single root latent detectors using local features without additional context). The individual LQP features $Disk_5^{3*}$ and $HVDA_7^{3*}$ outperform both LTP and HOG, and even their combination LBP+LTP+HOG.

Treating each class as an independent binomial trial, if a new feature has better AP than an old one on 8, 9, 10 of the ten classes, the hypothesis that the new feature is no better than the old one can be rejected with respectively 94.6%, 98.9%, 99.9% significance, irrespective of the actual differences in the AP's. By this criterion, 100 word and 150 word $Disk_5^{3*}$ outperform LTP at respectively the 98.0% and 98.9% confidence levels and LBP+LTP+HOG at the 82.9% and 96.9% ones on VOC2006. Without multiple trials or paired testing, standard deviation based tests are too weak to establish high levels of significance for the individual classes – VOC2006 has 233–1153 test examples per class giving the estimated AP's binomial-law standard deviations of 1.4–3.3% – but the corresponding Mean AP's have standard deviations of about 0.7% which is enough to establish 95% significance for 150 word $Disk_5^{3*}$ over LTP, and strong suggestiveness for it over LBP+LTP+HOG.

Finally, as with other local pattern features, combining LQP with HOG leads to significant performance improvements on VOC2006 – *c.f.* Table 7. As expected, the largest improvements occur for the weaker types of LQP features, with more modest improvements for the stronger ones.

4.3 Discussion

Given the above results, several points seem clear. Firstly, the fact that splitting ternary codes into their two binary halves leads to little performance loss both validates the use of splitting in LTP (and of binary coding in LBP), and suggests that the performance improvements provided by LQP are due mainly to the increased pattern sizes that lookup-table based coding permits, not per se to the absence of splitting or to the replacement of hand-specified codings with adaptive k-means ones. It also suggests that coding orders higher than ternary will give only limited further improvements and that they should be handled by splitting, but these points remain to be tested. Secondly, in agreement with the forms of existing local pattern features, patterns that sample pixels densely in a compact local neighbourhood around the centre seem to give the best performance, so disk-shaped ones are likely to be the best choice for many applications. Thirdly, even for large spatial supports, good results are obtained with quite modest codebook sizes – often even smaller than the corresponding LTP code. In any case, LQP can handle large codebooks with no loss of speed at run time – the issue is whether the subsequent classifier can handle the large histograms that result.

Despite the extra table lookup, LQP features remain very fast to train and test. For instance, HV_7^3, HVDA_7^{3*} and Disk_5^{3*} respectively take about 1.9, 3.1 and 4.5 seconds to scan a VOC2006 image. The extra time for Disk_5^{3*} is due to the need to resample pixels on circles, which takes much longer than the LQP table lookup. In comparison, replacing the LUT with a conventional explicit best codeword search over a 150 word codebook makes the Disk_5^{3*} feature computation 15 times slower, so that training and testing times become prohibitive, and larger codebooks are even slower. Even an efficent hashing-based coder doubles the overall LQP run time relative to LUT based coding.

5 Summary

We have presented Local Quantized Patterns (LQP), a generalized form of local pattern feature that replaces the traditional hand-built codebook reductions with vector quantization, using precompiled lookup tables to make coding very fast at run time. LQP inherits some of the flexibility and robustness of visual word representations without sacrificing the efficiency of traditional local pattern features. It can handle significantly larger patterns than previous local patterns (around three times as many pixels), and it easily adapts to different spatial topologies, quantization levels and datasets. These properties allow LQP to outperform traditional local pattern features such as LBP and LTP, and also well-established feature sets like HOG, on a range of challenging texture and object recognition datasets. In practice, patterns with dense sampling over compact spatial supports and split ternary coding work best, with the resulting LQP feature Disk_5^{3*} outperforming LBP, LTP, HOG and even their combination LBP+LTP+HOG on the INRIA Person and VOC2006 datasets. Codebook (and hence descriptor vector) sizes remain modest, and codebook learning is fast owing to the lookup table architecture.

Future work. So far we have tested only a small selection of the possible LQP configurations, and only on object detection and texture classification. LQP is also likely to be useful in many other applications such as face recognition, semantic segmentation, so much remains to be done.

Acknowledgements

This work was supported by grants from Higher Education Commission (HEC) of Pakistan and European research project CLASS.

References

1. Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE TPAMI (2002) 1, 2
2. Ahonen, T., Hadid, A., Pietikainen, M.: Face description with local binary patterns: Application to face recognition. IEEE TPAMI **28** (2006) 1
3. Pietikinen, M., Hadid, A., Zhao, G., Ahonen, T.: Computer Vision Using Local Binary Patterns. Springer (2011) 1, 2, 4
4. Tan, X., Triggs, B.: Enhanced local texture feature sets for face recognition under difficult lighting conditions. IEEE TIP **19** (2010) 1635–1650 1, 2
5. Chen, J., Shan, S., He, C., Zhao, G., Pietikainen, M., Chen, X., Gao, W.: WLD: A robust local image descriptor. IEEE TPAMI **32** (2010) 1705–1720 1, 6, 7, 9
6. Hussain, S., Triggs, B.: Feature sets and dimensionality reduction for visual object detection. In: BMVC. (2010) 112.1–112.10 1, 2, 7, 9, 11, 12
7. Leung, T., Malik, J.: Recognizing surfaces using three-dimensional textons. In: ICCV. (1999) 1010–1017 2
8. Csurka, G., Bray, C., Dance, C., Fan, L.: Visual categorization with bags of keypoints. In: Workshop on Statistical Learning in Computer Vision, ECCV. (2004) 1–22 2
9. Varma, M., Zisserman, A.: Classifying images of materials: Achieving viewpoint and illumination independence. In: ECCV. (2002) 255–271 2, 6
10. Tuytelaars, T., Schmid, C.: Vector quantizing feature space with a regular lattice. In: Proceedings of the 11th IEEE International Conference on Computer Vision, Rio de Janeiro, Brazil, IEEE (2007) 1–8 3
11. Cao, Z., Yin, Q., Tang, X., Sun, J.: Face recognition with learning-based descriptor. In: CVPR, IEEE (2010) 2707–2714 3
12. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: Brief: Binary robust independent elementary features. In: ECCV. (2010) 778–792 3
13. Elkan, C.: Using the triangle inequality to accelerate K-Means. In: ICML. Volume 20. (2003) 147–153 5
14. Moosmann, F., Triggs, B., Jurie, F.: Fast discriminative visual codebooks using randomized clustering forests. NIPS **19** (2007) 985 6
15. Moosmann, F., Nowak, E., Jurie, F.: Randomized clustering forests for image classification. IEEE TPAMI (2008) 1632–1646 6
16. Brodatz, P.: Textures: a photographic album for artists and designers. Volume 66. Dover New York (1966) 6
17. Valkealahti, K., Oja, E.: Reduced multidimensional co-occurrence histograms in texture classification. IEEE TPAMI **20** (1998) 90–94 6
18. Dana, K., Van Ginneken, B., Nayar, S., Koenderink, J.: Reflectance and texture of real-world surfaces. ACM Transactions on Graphics (TOG) **18** (1999) 1–34 6
19. Caputo, B., Hayman, E., Mallikarjuna, P.: Class-specific material categorisation. In: ICCV. Volume 2., IEEE (2005) 1597–1604 6

20. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. (2005) 886–893 7, 11, 12
21. Everingham, M., van Gool, L., Williams, C., Zisserman, A.: PASCAL Visual Object Classes Challenge results. http://www.pascal-network.org/challenges/VOC/voc/ (2006) 7
22. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. IEEE TPAMI (2009) 7
23. Joachims, T.: Making large-scale SVM learning practical. In Schlkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods - Support Vector Learning. The MIT Press, Cambridge, MA, USA (1999) 7
24. Papageorgiou, C., Poggio, T.: A trainable system for object detection. IJCV **38** (2000) 15–33 9
25. Viola, P., Jones, M.J.: Robust real-time face detection. IJCV **57** (2004) 137–154 9