Acta Cybernetica **22** (2015) 21–33.

Improved QR Code Localization Using Boosted Cascade of Weak Classifiers

Péter Bodnár* and László G. Nyúl*

Abstract

Usage of computer-readable visual codes became common in our everyday life at industrial environments and private use. The reading process of visual codes consists of two tasks: localization and data decoding. Unsupervised localization is desirable at industrial setups and for visually impaired people. This paper examines localization efficiency of cascade classifiers using Haar-like features, Local Binary Patterns and Histograms of Oriented Gradients, trained for the finder patterns of QR codes and for the whole code region as well, and proposes improvements in post-processing.

1 Introduction

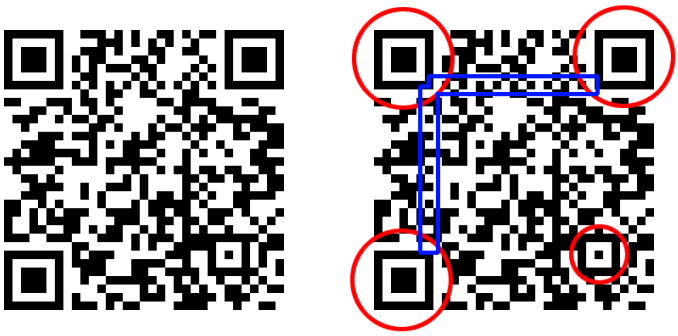
QR code is a common type of visual code format that is used at various industrial setups and private projects as well. Its structure is well-defined and makes automatic reading available by computers and embedded systems (Fig. 1). These codes can encapsulate significantly more embedded data than their one-dimensional ancestors referred to as bar codes. QR codes had a decent increase of usage in the last couple of years, more than other patented code types, like Aztec codes or Maxicodes. Furthermore, it has a well-constructed error correction scheme that allows recovery of damaged codes up to cca. 30 % of damage.

Image acquisition techniques and computer hardware have also improved significantly, that made automatic reading of QR codes available. State of the art algorithms do not require human assistance and assumptions on code orientation, position and coverage rate in the image [6, 9] any longer. However, image quality and acquisition techniques vary considerably and each application has its own requirements for speed and accuracy, making the task more complex.

The recognition process consists of two tasks: localization and decoding. The literature already has a wide selection of papers proposing algorithms for efficient QR code localization [1, 3, 7, 10, 11], however, each has its own strengths and weaknesses. For example, while those methods are proven to be accurate, morphological

*Department of Image Processing and Computer Graphics, University of Szeged, E-mail: {bodnaar,nyul}@inf.u-szeged.hu

DOI: 10.14232/actacyb.22.1.2015.3



(a) (b)

Figure 1: (a): example QR code, (b): same code with finder patterns (red circle) and data density patterns (blue rectangle) indicated

operations, convolutions, corner detection can be a bottleneck for processing performance. In order for the decoding to work well, an efficient localization step is required, which outputs boundary information of the code candidates as tightly as possible. After that, correction of blur, noise reduction and inverse perspective distortion takes place in the selected regions of interest (ROI), which operations can be considered as preprocessing for the decoding task. This paper pays attention only to the localization step, since while decoding can be time-consuming, it also can be considered straightforward using tight-fitting ROIs of an efficient locator.

Belussi et al. [1] built an algorithm around the Viola-Jones framework [12], which proved that, even though the framework was originally designed for face detection, it is also suitable for QR code localization, even on low resolutions. The authors used a cascade of weak classifiers, trained on the finder patterns (FIP) of the QR code. In the next section, their original idea is extended and examined, and results are discussed. We propose alternatives of both choosing the feature type of classifiers and their training target.

2 Localization Using Cascade Classifier Training

Using boosted cascade of weak classifiers is a common approach in general classification problems. A single classifier can be trained quickly, however, it will have low classification power (typically with high false negative rate), and thus it is considered weak. To overcome this issue, weak classifiers are chained, so the first classifier gets the original input, and each consecutive one has its input from the output of the preceeding one. If all classifiers have a high hit rate (typically from $0.990-0.999$) and a moderate false positive rate (around 0.5), the overall hit rate of the cascade is the product of the hit rates of all weak classifiers, and false positive rate is calculated in a similar manner. Using this approach, it is possible to train classifiers with high overall classification power, but without the need of complex

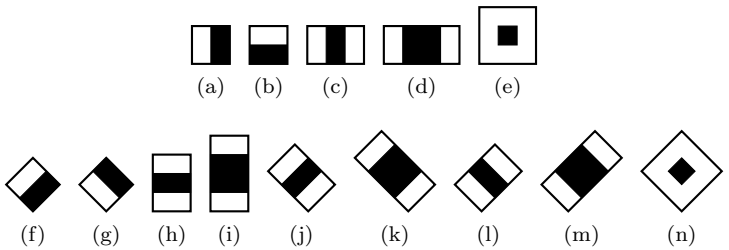


Figure 2: Haar-like features: edge type (a, b), line type (c, d), center type (e), and rotated entities (f–n)

features. Furthermore, each link in this chain can be composed of multiple weak classifiers, organized to a decision tree, or using Boosting methods.

2.1 Features for Object Recognition

In image processing, Viola and Jones [12] introduced the use of Haar-like features as the core of these weak classifiers, whose got their name from their similarity to the square-shaped functions of the Haar wavelet family. There are three sets of features, edge-type, line-type and center-type, and each set has its 45-degree rotated extension (Fig. 2), proposed by Lienhart et al. [5].

Each classifier has one or more features, defined by wavelet type, scale and orientation within the image region of interest (ROI). The classification process is the evaluation of these weak classifiers assembled in a cascade way, using a sliding window. The process is repeated on more than one scales, so a trained cascade can be used to detect objects of equal and larger size than they were present in the training database. Recurrences of a detected object are often filtered by grouping the overlapping results of different scales. Furthermore, we used Gentle AdaBoost in order to increase accuracy. Feature evaluation time can be further reduced by using integral images (Fig. 3) for the evaluation, that is derived from the original image as

$$I_{\text{int}}(x, y) = \sum_{u<x, v<y} I_{\text{orig}}(u, v), \tag{1}$$

where I_{int} and I_{orig} denotes for the integral image and the original image, respectively. Intensity sum of a rectangular region can be computed by accessing the sum values at the corner points of the rectangle in the integral image, as

$$\sum_{\substack{A_x \leq x < C_x \\ A_y \leq y < C_y}} I_{\text{orig}}(x, y) = I_{\text{int}}(C_x, C_y) - I_{\text{int}}(B_x, B_y) - I_{\text{int}}(D_x, D_y) + I_{\text{int}}(A_x, A_y), \tag{2}$$

where A, B, C and D denote for the corner points of the rectangle (Fig. 3).

Instead of Haar-like features, Local Binary Patterns (LBP) and Histograms of Oriented Gradients (HOG) can also be used for the feature evaluation. A paper on bar code localization [2] proposes partitioning of the image, and reading each block

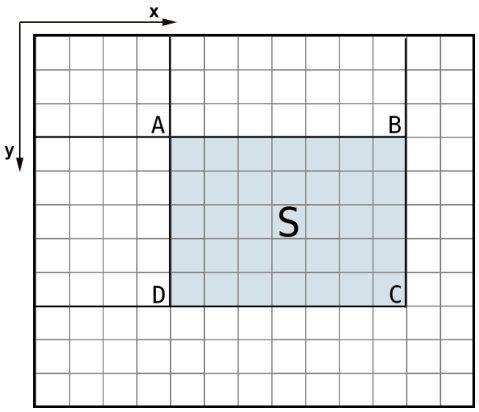


Figure 3: Calculation of the intensity sum over rectangle S. Intensity sum of S is calculated as $\text{Rect}(O,C) - \text{Rect}(O,B) - \text{Rect}(O,D) + \text{Rect}(O,A), O = \{0,0\}$ Integral image already has intensity sums calculated from O to points A, B, C, D, thus simplifying intensity sum query to $\mathcal{O}(1)$ w.r.t. rectangle size.

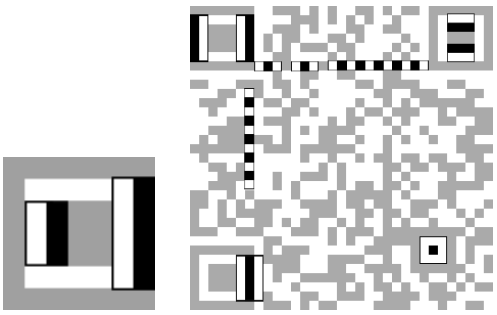
in a circular pattern. A 1D feature vector is formed this way, which makes a feature of bar code presence within the block. This concept is analogous to LBP [8], with the main difference of not using the center point for making the feature.

HOG descriptors were first introduced in pedestrian detection [4], however, it is often used in areas of computer vision where LBP, SIFT or shape context is applicable. There are some special cases [13] where LBP and HOG can be used together with improved overall accuracy, too.

2.2 Localization Based on FIPs and Whole Object

A classifier based on a Haar-like feature set is already discussed in the literature [1], and will serve as a reference method to our further experiments. The basic idea of QR code localization is the quick localization of possible FIPs in the image with high hit rate, then aggregation of the FIP candidates to FIP triplets of a possible QR code. FIP candidate localization is based on the cascade of boosted weak classifiers using Haar-like features, while the decision on a FIP candidate to be kept or dropped is decided by a geometrical constraint on distances and angles with respect to other probable FIPs.

While Haar-like feature based classifiers are the state of the art in face detection, the training process is more difficult on FIPs. A face has more, empirically observable, strong features, like the shape, position and shade of eyes, eyebrows, forehead, mouth, etc., than a QR FIP, which has only one prominent Haar-like feature to be perfectly covered with (Fig. 4(a)). In order to increase the strong features of the object intended to detect, we propose training of a classifier for the whole code area. Even though QR codes have high variability on the data region, they contain data density patterns, a fourth, smaller FIP that can be perfectly covered with the



(a) (b)

Figure 4: (a): FIP with two instances of a Haar-like feature. The feature fits for both the inner and outer black regions in all directions, however, this is the only feature that perfectly fits to the FIP. (b): Examples of Haar-like features fitting on a QR code, using FIPs and data density pattern.

center-type Haar-feature, furthermore, they contain the three discussed FIPs at the corners of the ROI (Fig. 4(b)). This approach also suggests promising rate of false positives due to the offered features, and is reassured at the Results section.

LBP and HOG based classifiers also can be trained both to FIPs and whole code areas, and since they are also considered fast and accurate general purpose object detectors, evaluation of their performance on code localization is highly motivated. Furthermore, LBP can be more suitable than Haar classifiers, since it is not restricted to a pre-selected set of patterns, while HOG can also be efficient due to the strict visual structure and limited number of distinct gradient directions of the QR code.

2.3 Classifier Training

Cascade classifier training starts with generating the database of labeled occurrences of the desired object to detect. The database can consist of arbitrarily acquired images, however, it has to be large enough for the training, with thousands of samples. A training database is often generated with one or a couple of positive samples rendered onto negative ones with various artificial modifications of the original object, like distortion, opacity changes, or addition of noise. As the next step, the database is divided into a larger and two smaller portions for training, validation and testing purposes. The training portion will be the input of the classifier training, while the test set will help to evaluate the performance of the training. `OpenCV` provides highly customizable classifier training as part of the library, with the Viola-Jones Haar-like features extended with the rotated entities of Lienhart et al., as well as LBP and HOG-based training. It offers feature symmetry, cascade and tree topology, and parameters for sample size, number of stages, splits, acceptance rate, and false positive rate as well.

The default weak classifier parameter values for true positive rate (TPR, recall)

and false positive rate (FPR) are 0.995 and 0.5, respectively, which means 99.5 % of the positive samples are classified correctly at each stage. A stage is a set of weak classifiers based on a single feature or a they can be Classification And Regression Trees (CART) themselves, with a given number of maximum splits. Classifiers based on simple features are boosted by one of the four offered boosting algorithms. To avoid combinatorical explosion of the parameter tuning, Belussi et al. [1] experimented with individual parameter variation while keeping other parameters fixed, and documented their empirically observed optimal parameter values for the Haar-training on FIPs. We have set the number of stages to 10, according to the experiments of [1]. For the first four stages, using only one feature was sufficient to reach the TPR and FNR defined above, while in later stages, more features were required, from 9 up to 15. The training did not contain a priori information about which features to prefer, they were chosen empirically as it is implemented in the OpenCV library. We trained a total number of six classifiers, based on Haar-like features, LBP and HOG, both for FIPs and full code objects. For the FIPs, feature symmetry is also recommended to speed up the training process, while usage of the rotated features of Lienhart et al. is not very useful, since these classifiers are not flexible enough to detect QR codes of any orientation. However, this issue can be solved by training two classifiers, for codes with orientation of $0°$ and $45°$, respectively. We used a 32×32 sample size, which is larger than the one of the reference method, since training to the whole code object requires finer sample resolution. We decided on the cascade topology for the classifier instead of a tree, since it showed higher overall hit rate in [1], and left required hit rate and false positive rate at the default values for each stage, with a total number of 10 stages. We trained our classifiers on a synthetic database consisting of 10 000 images. Images of the database are artificially generated QR codes, each containing a permutation of all lower- and uppercase letters and numerals, rendered with perspective distortion on to images not having QR codes. During the selection of the applied transformation matrices, we used such that shift the FIP not more than one FIP width, which property is needed for the assumption of maximum expected distortion at the postprocessing step of the FIP-based classification. However, this limit is large enough to render FIP-based classifiers unreliable. After that, Gaussian smoothing and noise have been gradually added to the images. The σ of the smoothing Gaussian kernel fell into the $[0, 3]$ range. For adding noise, a random image (I_n) was generated with intensities ranging from $[-127, 127]$ following normal distribution. This image was added gradually to the original 8-bit image (I_o) as $I = \alpha I_n + (1-\alpha) I_o$, with α ranging $[0, 0.5]$. The noise was added to the image using saturation arithmetic, i.e. values falling beyond the $[0, 255]$ range were clamped to the appropriate extreme intensities. Some samples with parameters being in the discussed ranges are present in (Fig. 5).

2.4 Post-processing

For the classifiers trained to FIPs, post-processing is needed to reduce the amount of false detections. Belussi et al. proposes searching through the set of FIP candidates

(a) $\sigma = 0.07$, $\alpha = 0.39$ (b) $\sigma = 0.65$, $\alpha = 0.77$ (c) $\sigma = 1.32$, $\alpha = 0.05$

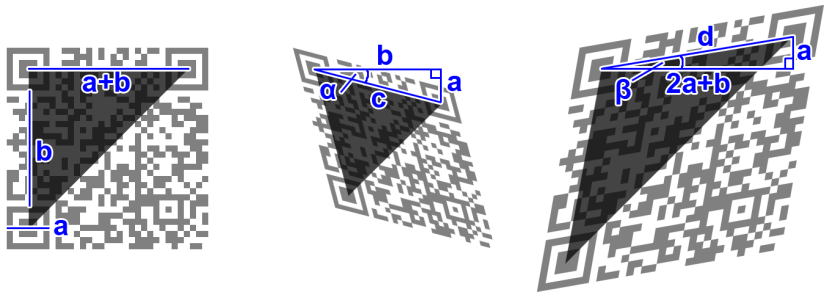
(d) $\sigma = 1.88$, $\alpha = 0.95$ (e) $\sigma = 2.86$, $\alpha = 0.65$ (f) $\sigma = 2.99$, $\alpha = 0.69$

Figure 5: Samples of the training database with different amount of smoothing and noise.

for triplets that can form QR code, using geometrical constraints. Since real-life images of QR codes also suffer perspective distortion, it is obligatory to give tolerance values for positive triplet response. We had to make assumptions on the geometry of the expected codes with respect to the distance of FIPs and the angle they enclose. In our case, 62 bytes of information are embedded into each QR code, which results in 33:7 as Code:FIP width ratio (Fig. 6(a)).

To the synthetic image set, we added perspective distortions that were capable of shifting the FIPs of the QR code by one FIP width at most in inward (Fig. 6(b)) or outward (Fig. 6(c)) direction from the code center. Let a be the FIP width and b the distance of the outer edges of two FIPs. For a code with no distortion, $a + b$ is the distance of the two other FIPs to the upper left FIP of the code, and their enclosed angle is 90° looking from the upper left (Fig. 6(a)). A QR code having a distortion that warps the FIP center inward by a (Fig. 6(b)), can be detected by letting $T_d = c/(a + b)$ tolerance to FIP distance, where $c = \sqrt{a^2 + b^2}$. Calculating with $(a + b) : a = 33 : 7$, the formula gives 0.7788 for T_d, which shows that letting 22.12 % of tolerance to the expected code size can detect codes up to the discussed distortion. The expected enclosing angle is $90 \pm 20.22°$, calculated by $T_a = \tan^{-1}(a/b)/90$, which is a 22.47 % of tolerance. The other case of distortion (Fig. 6(c)) can be calculated in a similar manner, and results in $T_d = 0.7707$

and $T_a = 0.1331$. According to these results, the post-processing step of triplet formation has to have a tolerance set to 23 % for FIP candidate distance and also for enclosing angle in order to not to lose any successfully localized QR codes during that step. Since detected FIPs are of different sizes, it would be possible to add a new constraint to the triplet formation defined as a tolerance factor for FIP size differences among triplets. However, due to the perspective distortions, it is not possible to narrow down results by FIP size variability, it only causes decreased hit rate. Furthermore, even with those relatively small degrees of distortion, necessary tolerances for distance and angle are high enough to compromise the filtering power of the triplet formation rule.



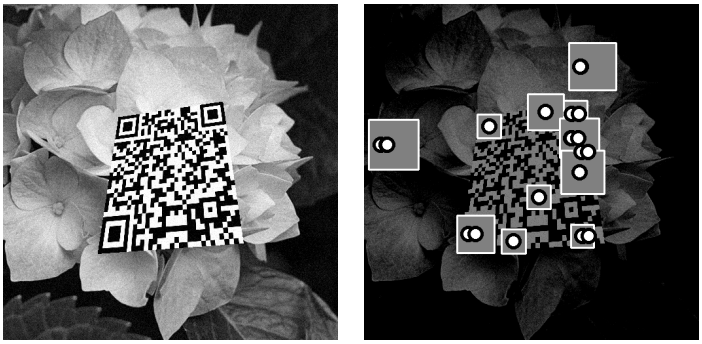
(a) (b) (c)

Figure 6: Example for deciding on triplet formation tolerance. From the top left corner of a perfect QR code, the other two FIPs are enclosing 90 degrees and distance of FIP centers is $a + b$ (a); Considering two scenarios of distortion where FIPs are shifted inwards (b) and outwards (c) with a, distances and angle tolerances for acceptance can be calculated using basic geometry.

Belussi et al. [1] proposed similar constraints for FIP triplet formation. According to their paper, each FIP candidate center is a vertex, which has a size attribute. Their defined vertex size equals to FIP width, which is the same as FIP height since FIPs are square shapes. Distance of the vertices are limited to 18 times vertex size for successful triplet formation, and a tolerance of 25 % for vertex size is applied within each triplet. As a final filter of the triplets, each one has some angle and distance constraints. For successful triplet formation, all these three rules have to be met. However, this still requires the calculation of angles and distances for all FIP candidates.

While classifiers trained to the whole code area need no post-processing, FIP-trained ones require the formation of a distance matrix for all FIP candidate pairs, and a direction matrix that stores the angle of the line segment defined by all FIP pairs. After that, reading through n FIP candidates still takes $\mathcal{O}(n^3)$ time, which is a bottleneck since a FIP-trained classifier can produce a large number of FIP candidates (Fig. 7).

Originally, FIPs are designed to indicate QR code presence while scanning the



(a) (b)

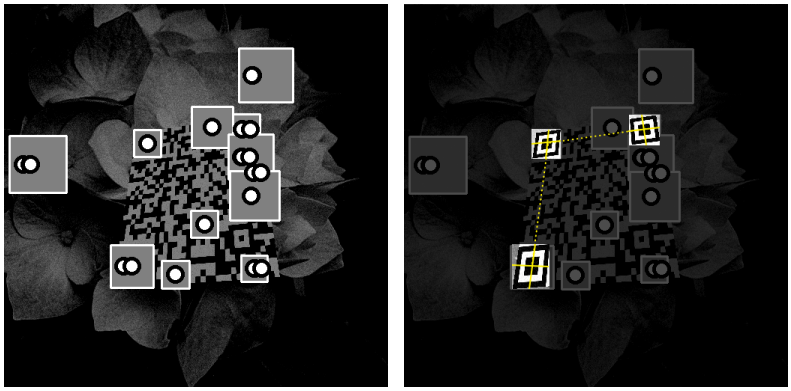
Figure 7: Example QR code and result of a Haar classifier trained on FIPs. Original image (a); feature image (b) with numerous FIP candidates (gray square), and marked candidates (white circle) that have passed post-processing. FIP-based classifiers show high false positive rate that neither the triplet formation constraint can reduce. Circles on the same square mean that a FIP candidate is participating in formation of more than one probable QR code.

image line by line. A FIP has a binarized intensity runlength profile of 1-1-1-3-1-1-1 when scanned horizontally, vertically, or diagonally. That approach involves reading the whole image, which is a slow procedure, and it is sensitive to noise and blur. Cascade classifiers are designed to overcome these issues, however, the concept of scan-lines can be re-introduced within the FIP candidates as a powerful filtering step before the triplet calculation. Even though the FPR of FIP candidates is about 0.5, the overall proportion of FIP size to image size is small enough to perform scan-line analysis. A maximum of four scan-lines with the step of 45° is sufficient to determine if there is a FIP candidate present in the box that the classifier outputs. If less than two of them give positive intensity profile response, that FIP candidate can be dropped. Furthermore, the scan-lines of positive response give hints about the direction of neighboring FIPs for triplet formation, although, in most cases, tracing those hints in image space would be slower than iterating through the remaining FIP candidates. However, after the triplets have been formed, the orientation of the scan-lines can serve as a final constraint for triplet validation (Fig. 8).

As further post-processing, rotated bounding boxes can be computed, which are more tight-fitting in most cases than axially aligned ones. This is the final step of the localization task. Skew correction of the ROI, inverse perspective transformation and binarization are considered preprocessing steps of the decoding task.

3 Evaluation and Results

The classifiers have been trained using `OpenCV` on the discussed training database. Training time for Haar features took cca. 15 hours on a Core 2 Duo 3.00 GHz



(a) FIP candidates (b) Scan-lines and final validation

Figure 8: Scan-line postprocessing of the FIP candidates in order to reduce false positives. FIP candidates are scanned with four lines, and dropped if less than half of the lines have positive response. As a final validation, the angle of positive scan-lines can be compared between FIP candidates within the triplets. For a valid triplet, positive scan-lines have similar angles.

CPU, while LBP training took about 1.5 hours, and HOG-training was the fastest, taking only about 30 minutes. There were only minor increases in training times of each category when training target was the full code object instead of the FIPs. Processing of test images with the trained classifiers has no significant difference respecting detection time, and each one is fast enough for real-time application. Detection time mostly depends on the scaling parameter in multi-scale detection. The default scaling factor is 1.1 in OpenCV, in which case detection takes cca. 100–200 ms for 512×512 px images on an Intel Core 2 Duo 3.00GHz CPU.

Table 1 shows performance measures of the examined cascade classifiers. HAAR-FIP, as stated by authors of [1], has a hit rate above 90 %, and represents a good solution for FIP-training. However, all FIP-based classifiers have poor precision compared to the ones trained for full code region, and they can cause serious overhead for the next, decoding step of the QR code recognition process. Classifiers based on LBP and HOG do not reach the hit rate of the one with Haar features. HOG-FIP shows a noticeably higher precision than its siblings, but still cannot be considered as an effective classifier according to its hit rate. Performance measures are made by a 90 % minimum required overlap of detected bounding box to the ground truth for a true positive.

For classifiers with the whole code object as their target, results are much more spectacular. Both HAAR-FULL, LBP-FULL and HOG-FULL show outstanding hit rate and acceptable precision. The LBP-FULL classifier was able to detect all codes of the test database with a very low amount of false positives, having an F-measure over 0.95.

Table 2 shows results of the trained classifiers for the public database of Sörös

	Precision	Hit rate	F-measure
HAAR-FIP [1]	0.1535 ± 0.0920	0.9436 ± 0.0753	0.2640 ± 0.1125
LBP-FIP	0.1686 ± 0.0530	0.7356 ± 0.1112	0.2743 ± 0.0773
HOG-FIP	0.4753 ± 0.2466	0.7885 ± 0.1960	0.5931 ± 0.1947
HAAR-FULL	0.4208 ± 0.2404	0.9995 ± 0.1092	0.5923 ± 0.1050
LBP-FULL	0.9050 ± 0.1312	0.9999 ± 0.0857	0.9501 ± 0.0721
HOG-FULL	0.5390 ± 0.2549	0.9975 ± 0.1001	0.6999 ± 0.1221

Table 1: Test results of the proposed cascade classifiers based on Haar-like features, LBP and HOG, both trained for finder patterns (-FIP) and whole code objects (-FULL).

	Precision	Hit rate	F-measure
HAAR-FULL	0.2366 ± 0.2325	0.9060 ± 0.2192	0.3752 ± 0.1285
LBP-FULL	0.3663 ± 0.3265	0.7607 ± 0.1847	0.4944 ± 0.1430
HOG-FULL	0.7817 ± 0.2842	0.9487 ± 0.2871	0.8571 ± 0.2141
HAAR-SOROS	0.9999 ± 0.4220	0.7619 ± 0.2587	0.8649 ± 0.2937
LBP-SOROS	0.3684 ± 0.2082	0.9999 ± 0.1640	0.5385 ± 0.0973
HOG-SOROS	0.9999 ± 0.2127	0.9524 ± 0.1063	0.9756 ± 0.1347

Table 2: Classifier performances for the database of Sörös et al. [10]. The ones ending with -FULL are the same classifiers trained on our synthetic database, while -SOROS classifiers are trained on their public database.

et al. [10]. HAAR-FULL, LBP-FULL and HOG-FULL are the same classifiers like in Table 1, they are trained only in our training database and were evaluated with no modifications. The last three classifiers, HAAR-SOROS, LBP-SOROS and HOG-SOROS are classifiers using full code object, trained on their database which consists of about 100 arbitrarily acquired images taken with iPhone camera. The main difference between the two databases besides one containing synthetic data and the other real, is the higher variability in size and orientation of QR codes for the latter. As expected, each classifier has noticeably lower hit rate, since they were trained using another database with different constraints, however, results still prove that cascade classifiers are a reasonable approach for the selected task, even when they are evaluated on a significantly different test set.

We also experimented with training cascade classifiers on the Sörös data set, however, training had only 85 samples as input and 21 for evaluation, which is too few for making strong statements in a machine learning context. HAAR-SOROS and HOG-SOROS had no false positives at all, but they were also unable to detect all instances. LBP could be trained well for the database with respect to hit rate, but probably due to the low count of training samples, shows poor precision.

In conclusion, the most efficient classifier disposes of the following parameters: LBP of 32×32 sample size used for feature extraction in cascade topology, boosted by Gentle AdaBoost, and a 10 stage learning phase with 0.995 hit rate and 0.5 false

alarm rate, with no splits or tree structure. In cases where orientation variability is high for the expected codes, we recommend training two separate LBP-FULL classifiers with two training sample databases, with sample orientations around 0° and 45°, respectively.

4 Concluding Remarks

QR codes became common for the past few years and their wide use made automatic reading desirable. We presented various cascade classifiers based on different features and training target, and studied their performance and capability for QR code localization. Our approach can be used in real-time applications with high hit rate and a moderate false positive rate that depends mainly on training parameters that can be tuned to meet the requirements of each final application. Efficient automatic localization of visual codes is desirable at many industrial setups and also for end-user cases, where localization is performed using only little human assistance, like on smartphones used by visually impaired people.

According to our experiments, cascade classifiers seem to be a decent option for QR code localization, especially a classifier using LBP for features and trained for the whole code object.

References

[1] Belussi, Luiz F. F. and Hirata, Nina S. T. Fast QR code detection in arbitrarily acquired images. In *Graphics, Patterns and Images (Sibgrapi), 2011 24th SIBGRAPI Conference on*, pages 281–288, 2011.

[2] Bodnár, Péter and Nyúl, László G. A novel method for barcode localization in image domain. In *Image Analysis and Recognition*, volume 7950 of *Lecture Notes in Computer Science*, pages 189–196. Springer Berlin Heidelberg, 2013.

[3] Chu, Chung-Hua, Yang, De-Nian, Pan, Ya-Lan, and Chen, Ming-Syan. Stabilization and extraction of 2D barcodes for camera phones. *Multimedia Systems*, 17:113–133, 2011.

[4] Dalal, N. and Triggs, B. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1, June 2005.

[5] Lienhart, Rainer, Kuranov, Alexander, and Pisarevsky, Vadim. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *Pattern Recognition*, volume 2781 of *Lecture Notes in Computer Science*, pages 297–304. Springer Berlin Heidelberg, 2003.

[6] Lin, Jeng-An and Fuh, Chiou-Shann. 2D barcode image decoding. *Mathematical Problems in Engineering*, 2013.

[7] Ohbuchi, Eisaku, Hanaizumi, Hiroshi, and Hock, Lim Ah. Barcode readers using the camera device in mobile phones. In *Cyberworlds, 2004 International Conference on*, pages 260–265, 2004.

[8] Ojala, T., Pietikainen, M., and Harwood, D. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision and Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, pages 582–585 vol.1, Oct 1994.

[9] Parikh, Devi and Jancke, Gavin. Localization and segmentation of a 2D high capacity color barcode. In *Applications of Computer Vision, 2008. WACV 2008. IEEE Workshop on*, pages 1–6, 2008.

[10] Sörös, Gábor and Flörkemeier, Christian. Blur-resistant joint 1D and 2D barcode localization for smartphones. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, MUM '13, pages 11:1–11:8, New York, NY, USA, 2013. ACM.

[11] Szentandrási, István, Herout, Adam, and Dubská, Markéta. Fast detection and recognition of qr codes in high-resolution images. In *Proceedings of the 28th Spring Conference on Computer Graphics*, SCCG '12, pages 129–136, New York, NY, USA, 2013. ACM.

[12] Viola, Paul and Jones, Michael. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001.

[13] Wang, Xiaoyu, Han, T.X., and Yan, Shuicheng. An HOG-LBP human detector with partial occlusion handling. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 32–39, Sept 2009.