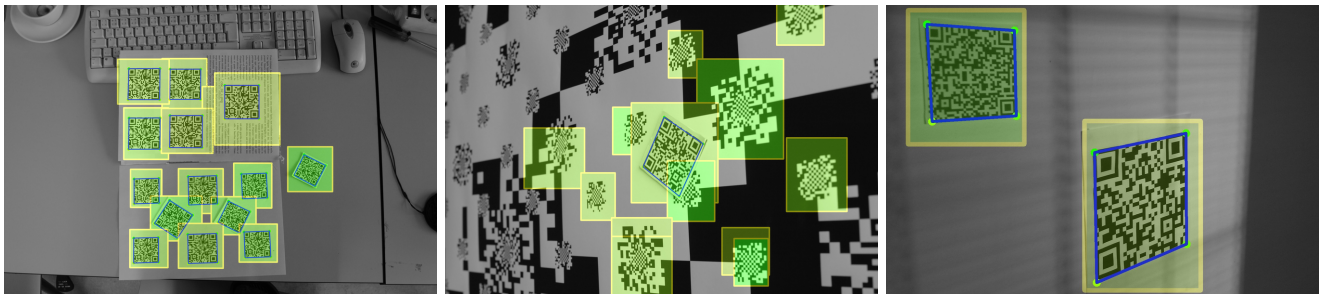
Fast Detection and Recognition of QR codes in High-Resolution Images

István Szentandrási* Adam Herout† Markéta Dubská‡
Graph@FIT, Brno University of Technology



Abstract

This paper deals with detection and recognition of matrix codes, such as the QR codes, in high-resolution images of real-world scenes. The goal is to provide a detector capable of operation in real time even on high-resolution images (several megapixels). We present an efficient algorithm for detection of possible occurrences of the codes. This algorithm is characterized by a very low false negative rate and a reasonable false alarm rate. The results of our algorithm are to be followed by an accurate detection/recognition algorithm. We propose to use a recent matrix code detection and recognition algorithm based on Hough transform, because it can reuse some information computed by our new pre-detection algorithm and thus a further reduce of computational demands can be achieved. Since there are no publicly available annotated datasets for evaluation of this kind of algorithm, we collected a number of images and annotated them; these images will be made publicly available to allow for a proper comparison. Our algorithm was evaluated on this dataset and the results are reported in the paper.

CR Categories: I.5.4 [Pattern Recognition]: Applications—Computer vision;

Keywords: QR code detection, Histogram of Oriented Gradients

1 Introduction

QR codes [Denso Wave Inc. 2010] are a very popular case of matrix codes (or 2D barcodes). They are receiving an increasing popularity among smartphone users. The QR codes themselves or similar matrix codes (Aztec, DataMatrix, etc.) are being routinely used by different industries to carry meta-information attached to real-world objects. These matrix codes are designed so that they would be detected in images where they cover a significant portion of the image and where they are not deformed by perspective projection. However, their detection in high-resolution images of real-world complex scenes is desirable. These views would involve rotation, perspective deformation, and other distortions of the codes.

When the QR code covers a significant portion of the input image ($\approx 25\,\%$ or more) and only one code appears in the image, the situation is simplified and algorithms exist which solve this situation [Alfthan 2008], [Herout et al. 2012]. However, in complex scenes with more than one QR code skewed by perspective and rotation (see Figure 5 for an example), the process of detection and recognition of the codes is different. Alfthan [2008] observes that for "just" detecting the code, a lower-quality image and other means of processing can be used – compared to recognition of the code. This may lead to two distinct algorithms – an algorithm for detection of the codes in a complex high-resolution image, and an algorithm for recognition of individual codes in the identified sub-images.

Some researchers keep relying on the *Finder Patterns* embedded in each QR code (however absent in other codes, such as the DataMatrix) [Belussi and Hirata 2011], [Sun et al. 2007], [Ohbuchi et al. 2004]. Others detect the code as a whole – by using texture analysis [Hu et al. 2009], morphological operators [Arnould et al. 1999] and other techniques [Jain and Chen 1993]. Most of these approaches are computationally very demanding (they would use the Canny edge detector, several passes by a morphological operator, convolution with large kernels, and similar techniques). Even the algorithms targeting on real-time operation are not quite usable on images with high resolution (for example, a recent solution by Belussi and Hirata [2011] takes 50-150 ms to process a 640×480 image in one pass and several passes are needed).

This paper presents a new method for detection of the QR codes in complex scenes, based on Histograms of Oriented Gradients (HOG) [Dalal and Triggs 2005] and segmentation of image parts based on HOG. Our solution is computationally very efficient – only a fraction of image pixels need to be visited in order to detect the QR codes in a high-

*e-mail: iszent@fit.vutbr.cz
†e-mail: herout@fit.vutbr.cz
‡e-mail: idubska@fit.vutbr.cz

SCCG 2012, Budmerice, Slovakia, May 2 – 4, 2012.
© 2013 ACM 978-1-4503-1977-5/12/0005 $15.00

resolution image. Its byproduct are the evaluated HOGs for the individual detected QR codes; they can be used by our recent algorithm [Herout et al. 2012] for matrix code recognition, sparing additional computational resources.

The detection algorithm is evaluated on a dataset of 115 high-resolution (up to 15MPix) images containing one or more QR codes (251 code instances in total). Our algorithm successfully localized 95 % of all QR codes and for 80.5 % of all codes (localized or not) the binary map was successfully extracted. The required time to achieve these results was in average under 250 ms for 15 Mpix images.

The rest of the paper is structured as follows. Section 2 summarizes the necessary prerequisites for our algorithm. Section 3 outlines our recent algorithm for recognition of QR codes, based on the Hough Transform and analysis of vanishing points. Section 4 explains the algorithm presented in this paper. Section 5 describes the datasets obtained and collected for the evaluation and gives the experimental results.

2 Previous Work

The QR code was designed in such a way, so that it can be easily localized by finding the predefined structures at its three main corners (see Figure 1). There have been many attempts to speed up and improve the detection of the QR codes the way it was intended using the *Finder Patterns* (FIP). Ohbuchi et al. [2004] used scanlines on a preprocessed image to find FIPs and then applied an inverse perspective transformation for image reorganization to extract the binary

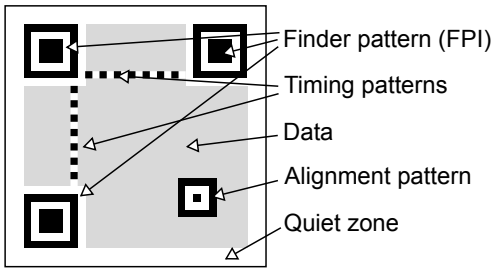


Figure 1: The structure of the QR code.

information enclosed in the QR code. Sun et al. [2007] used a similar approach, but for locating the FIPs they used the Canny edge detector and looked for square contours.

Choosing the correct threshold to binarize the image and find FIPs depends among others on the properties of the camera and illumination. Liu et al. [2008] used multi-level threshold to deal with varying lighting conditions for QR code detection.

Given the regular structures of QR codes and matrix codes in general, there has been research on using the Hough transformation for matrix code or barcode detection and localization. Muniz et al. [1999] presented a robust method for decoding linear barcodes using the Hough transform. Wang and Zou [2006] used the Hough transform to locate the four corners of the 2D bar code region and used the second derivative image in combination with bidirectional centripetal run-length to decode the bar code. Parikh and Jancke [2008] also used the Hough transform. They combined it with adaptive threshold and texture analysis to locate the correct position of 2D high capacity color barcodes and to decode them.

The matrix codes (including the QR codes) have a very specific texture characteristic. Thus the problem of localization of the QRcode is similar to image segmentation based on the texture features and then selection of a segment with requested characteristics. Methods of texture based segmentation vary in used textural descriptors such as the Local Binary Patterns [Ojala and Pietikäinen 1999], Gabor features [Weldon et al. 1996] or Markov random field statistic [Haindl and Mikes 2005].

Ouaviani et al. [1999] presented an image processing framework exploiting the regular structure of 2D barcodes. First they used the histogram of gradients to detect regions of interest in the image. To get the precise area of the code, they segmented out the region containing the code based on an average gray-level of pixels in a small neighborhood using region growing and found the smallest 4-sided polygons surrounding the segmented code.

3 Detection and Recognition of QR Codes by PClines

The algorithm is based on our recent algorithm for detection of chessboard grids and matrix codes [Herout et al. 2012]. It efficiently searches for straight lines that coincide with two dominant vanishing points by using the Hough transform and a line parameterization which is a point-to-line mapping. The algorithm processes the input image in four steps:

1. Extraction/selection of edges in the input image. The image is processed by a computationally cheap edge detector and information about the gradient for the edge pixels is obtained. A histogram of edge orientations is constructed during the extraction and two dominant edge orientations roughly 90° apart are selected out of all the edges (see Figure 2).

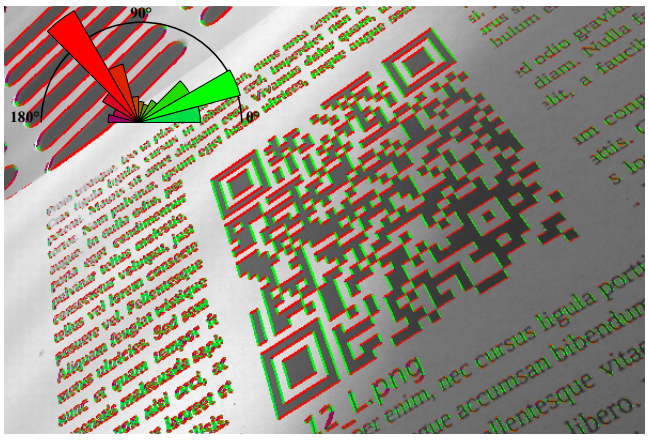


Figure 2: Edge extraction and selection. Original QRcode image with detected edges. In the left-top corner histogram of edge orientations – two peaks can be found and only the relevant edges are selected for subsequent use.

2. Accumulation to the Hough space. For each of the two dominant groups of edges, a small part of the com-

plete Hough space is being accumulated (see Figure 3). A recent line parameterization by Dubska et al. [2011] is used, because a vanishing point is there represented by a line in the Hough space (contrary to the θ-ϱ line parameterization [Duda and Hart 1972] typically used for line detection by the Hough transform). Since only a small fraction of the Hough space is accumulated, and only a selected subset of the edges extracted from the image is used, this step is fairly efficient.

3. Testing hypotheses about the vanishing points. Each of the two parts of the $\mathcal{TS}$ space is searched for the vanishing point and the corresponding group of projected parallel lines. First, N highest local maxima are found in the accumulator space (see Figure 3a). These maxima are used for generating hypotheses about the vanishing points and the corresponding groups of concurrent lines. One hypothesis is defined by 3 local maxima (i.e. triplets are drawn out of the N points) and it uniquely defines a spatial distribution of local maxima corresponding to the group of perspectively projected parallel lines. The confidence score for each hypothesis calculates the similarity of real and estimated maxima and minima.

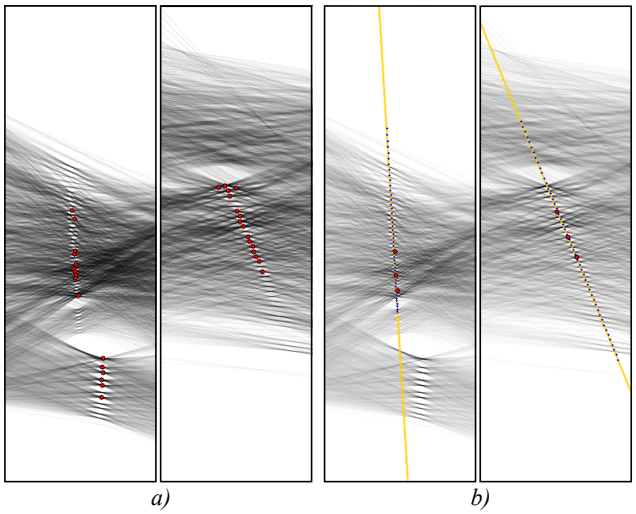


Figure 3: *a)* Two accumulation areas with 16 best found maxima. These 16 maxima are used for generating the hypotheses. *b)* One generated hypothesis for each half of the $\mathcal{TS}$ space. A line is approximated by 3 random maxima. The $\mathcal{TS}$ space is searched along the line and a score of the hypothesis is computed.

4. Extraction of the marker bitmap. For sampling the input image in the center of each module, the minima between two consecutive maxima are used (see Figure 3b). A Cartesian product of the two groups of minima in the Hough space determines individual modules of the marker. These modules are extracted and they form a low-resolution bitmap which is searched for the synchronization patterns and further processed (see Figure 4).

4 Rapid Detection of Regions with a Possible QR code

In this project we focused on relaxing the constraint of having exactly one 2D barcode in the image and consequently

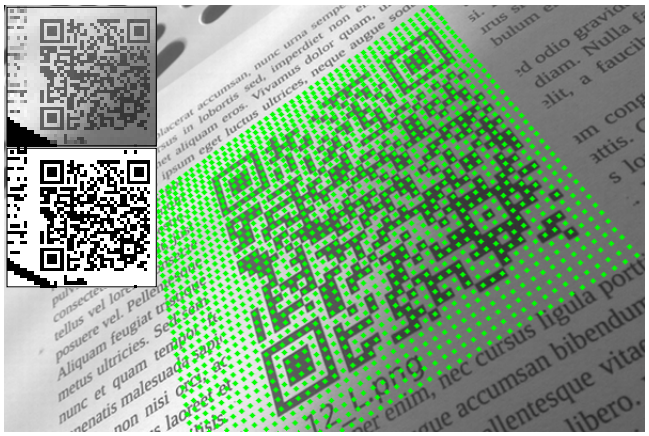


Figure 4: Extraction of the matrix code. The points shown in green are sampled and form a bitmap which is then adaptively thresholded.

allowing that barcodes cover only a small part of the image (see Figure 5 for an example). We propose an algorithm that can efficiently localize and extract candidate positions for the 2D barcodes even in high-resolution images. In order to achieve this, we exploit the property of 2D barcodes of having a regular distribution of edge gradients to get probable positions for full QR code detection.

Figure 5: The original 15MPix image. Multiple codes are present in the image; they are subject to perspective projection; dominant edges and text appear in the image.

4.1 Processing in Tiles

A naive way of finding the 2D barcodes without using special properties of specific barcode types in a high resolution image at every reasonable scale would be to scan through the image at each scale with a sliding window of a given size and run the detection algorithm (such as the one described in Section 3) at each position. This would require too much computational time.

2D barcodes, however, have some common properties that could be potentially exploited to help localizing them. Two most important of these are the regular distribution of edge

lines and a characteristic distribution of the gradient directions.

We propose separating the image into a regular grid of tiles. Each tile could serve as an indicator for a probability of being part of a 2D barcode. Let $\mathcal{T}(u, v)$ be a tile at index $u, v \in \mathbb{N}_0$. The tile then corresponds to a square image region with top-left corner (uw, vw), where w is the width of the tile (all tiles share the same size). Figure 6 shows an example of such a grid. The overlayed green intensity represents the probability of the tile being possibly a part of a QR code.

Extraction of the Histogram of Oriented Gradients. To get an indicator for QR code presence, the algorithm first extracts the Histogram of Oriented Gradients (HOG) for each tile. For each pixel, the algorithm computes the size of the gradient: both its magnitude and orientation. Only gradients whose magnitude exceeds a given conservative threshold are processed (just as in Section 3). The threshold is set fairly low so that a reasonable amount of blur does not spoil the detection. The stored edge pixels are used to create the HOG for the tile. We used a histogram with a low number of bins (e.g. 12 bins); higher histogram resolution is futile due to the limitations of gradient direction precision. The resulting histogram is a vector for each tile: $\mathbf{h} = \mathbf{H}_n(\mathcal{T}(u, v))$, where n is the number of bins (see Figure 6 for HOG representation. The highlighted red and blue bins of the HOGs are the detected two dominant edge orientations.)

To speed up the process, the histograms can be approximated by taking only pixels along scanlines vertically and horizontally separated by a given number of pixels. By processing only the scanlines, only a small percentage of input image pixels have to be processed for each tile, but still have a precise-enough approximation of the HOG.

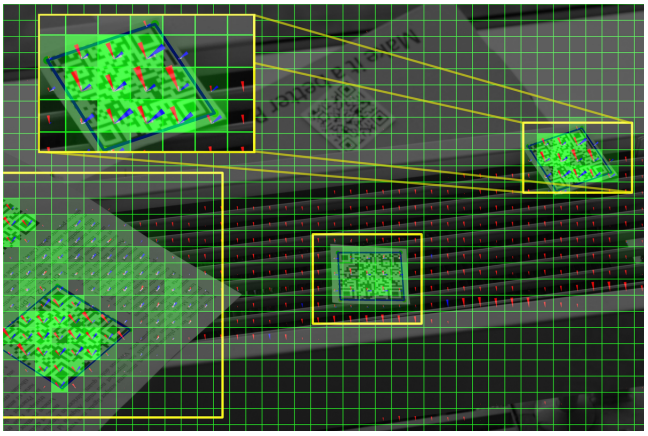


Figure 6: The grid with histograms of oriented gradients.

Feature Vector Extraction. In addition to the gradient histogram, a feature vector is computed for each tile. It contains the normalized histogram and four additional values: corresponding angles of the two main gradient directions in the histogram, the number of edge pixels per unit area and an estimation of the probability score of a chessboard like structure in the tile based on the histogram. Let then $\mathbf{v} = (p, \alpha_1, \alpha_2, \frac{N_e}{A}, \mathbf{h}_{norm})$ be the feature vector for a tile with normalized histogram $\mathbf{h}_{norm}$, pixel area A, and the number of edge pixels over a threshold N_e. The probability score is computed in the following way:

$$p = \left(1 - \frac{||\alpha_1 - \alpha_2| - \frac{\pi}{2}|}{\frac{\pi}{2}}\right) \frac{2\min(\mathbf{h}_a, \mathbf{h}_b)}{\mathbf{h}_a + \mathbf{h}_b}, \qquad (1)$$

where $\mathbf{h}_a$ and $\mathbf{h}_b$ are the values in the histogram corresponding to the two dominant edge orientations. The first term ensures that the two dominant directions are approximately $\frac{\pi}{2}$ apart; the second term prefers equally high peaks in the histogram.

The two main gradient directions detected from the HOG α_1 and α_2 meet an additional constraint that $|\alpha_1 - \frac{\pi}{2}| \ll |\alpha_2 - \frac{\pi}{2}|$ or $\alpha_1 < \alpha_2$. The order of the angles is important for the segmentation.

Segmentation of the Tiles. This vector of features can be used for segmentation of the tiles. For segmentation we currently use a simple 4-neighborhood blob coloring [Rosenfeld and Pfaltz 1966], which requires two passes for each level in the hierarchy of tiles. We used only the first angle α_1, the probability score p and the edge density $\frac{N_e}{A}$ for segmentation for performance reasons. The segmentation is done on a relatively small number of elements, so it does not introduce any major computational overhead.

The result of the segmentation (Figure 7) is a set of connected tiles $\{S_1, S_2, \ldots, S_k\}, k \in \mathbb{N}$. If $A(S_k)$ is the area of the axis-aligned bounding box in $pixel^2$ and then let the probability score that the segment is a possible position of a QR code be:

$$P(S_i) = \frac{1}{A(S_i)} \sum_{\mathcal{T} \in S_i} p\frac{N_e}{A}, \quad i \in \{0, .., k\} \qquad (2)$$

where p and $\frac{N_e}{A}$ are values from the feature vector for tile $\mathcal{T}$.

Segment Classification. Based on the computed score $P(S_i)$ and the shape of a set of tiles S_i we use a binary classifier to determine whether the area covered by a given segment should be further processed to find QR codes:

$$C(S_i) = \begin{cases} 1 & \text{if } P(S_i) \geq T \text{ (is a possible position)} \\ 0 & \text{otherwise (not a probable position)} \end{cases}, \qquad (3)$$

where T is an experimentally acquired threshold. The regions classified as probable positions of a 2D barcode in our example in Figure 6 are represented by thick yellow rectangles.

An example of the segmentation is shown in Figure 7. The edge length of the tiles in this case were $w = 120$ pixels. Two of four QR codes were precisely localized. The segment corresponding to the third, bigger one on the left side of the image contains also some parts of the text, since at this resolution it has properties similar to the QR code itself. The fact that the QR code candidate rectangle does not tightly fit the actual code edges does not prevent it from being detectable by the detection/recognition algorithm. The fourth blurred QR code in the background did not get localized due to the small number of edge pixels and the near uniform distribution of the gradient directions.

4.2 Multiscale Processing

The size of the tiles can have a major effect on the success of finding correct candidate positions. Choosing too large

Figure 7: The segments used for finding the QR code. Red color component encodes the edge count per tile (inverted). Green color component represents the angle of the first dominant direction (α_1). Blue color component indicates the absence of the QR code based on p (refer to Eq. (1)).

region size might cause overlooking small matrix codes. For example, when a small QR code surrounded by large noise covers only a small part of four neighboring tiles, the probability score of a regular structure in the four tiles would be very small. On the other hand, choosing too small tile size w would cover only a small part of a large QR code and the segmentation would fail to group the tiles covered by the QR code.

We propose using a quad-tree of tiles $\mathcal{T}_l(u,v)$, where l is the level. The HOG and edge extraction have to be computed only at the lowest level $l = 1$. The histogram of the tiles at a higher level in the hierarchy can be computed simply by accumulating the corresponding bins of the histograms of the child tiles:

$$\mathbf{H}_n(\mathcal{T}_l(u,v)) = \sum_{i=0}^{1}\sum_{j=0}^{1}\mathbf{H}_n(\mathcal{T}_{l-1}(2u+i, 2v+j)). \quad (4)$$

The segmentation and classification of the segments can be done at each level (or alternatively, a hierarchical segmentation can be used). Since the gradients and edge pixels are not recomputed for higher levels and the number of segments is relatively low, so this does not mean a significant computational overhead.

The whole algorithm for detection of QR codes in a high-resolution image is depicted in Algorithm 1.

Algorithm 1 QR code detection in high-resolution images

Input: Image I
Output: Detected QR codes

```
compute H_n(T_1(u,v)) by edge extraction
compute H_n(T_l(u,v)), l ∈ {2,...,l_max} from lower-level histograms (4)
for all l ∈ {1,...,l_max} do
    compute feature vectors v_l(u,v) from the histograms
    compute the segments S = {S_1, S_2, ..., S_k}, k ∈ ℕ
    for all S_i ∈ S do
        compute segment probability P(S_i) (2)
        if C(S_i) == 1 then
            run QR code detection algorithm
        end if
    end for
end for
```

5 Experimental Results

In order to evaluate the performance we collected a dataset of challenging real-life images. Since no standard dataset is publicly available for evaluation of QR code detection algorithms, we acquired the images ourselves and the dataset will be made publicly available[1]. The images in the dataset were taken both with a cell-phone camera and a professional high-definition camera. The resolution of the images scales from 1 MPix to 15 MPix. The cell-phone camera had a maximal resolution of 5 MPix, so larger images were all taken with the professional camera.

The images in the dataset contain several different QR codes. An image contains at least one QR code, the maximal number of codes in a scene was 14. The percentual image coverage of the codes varies from 2 % to 70 %. The position and size of the codes for all images was annotated. Additionally we also annotated some further properties/flags for the codes: rotation, perspective deformation, varying lighting conditions and blur.

We divided the evaluation into two parts. In the first part we evaluated only the performance of the candidate position search using our grid. In the second part we evaluated the performance of the whole algorithm including the 2D barcode detection method described in Section 3.

5.1 Candidate Position Search

In order to evaluate the efficiency and precision of the candidate search with our grid, we used 1-precision – recall graph proposed by Mikolajczyk et al. [Mikolajczyk and Schmid 2005]:

$$recall = \frac{\#correct\,matches}{\#all\,annotated\,positions}, \quad (5)$$

$$1 - precision = \frac{\#false\,matches}{\#correct\,matches + \#false\,matches}. \quad (6)$$

A candidate position is considered a successful match, if it contains exactly one known full QR code which covers at least 25 % of the candidate area. The correct matches also do not contain duplicates for the same QR code. The $\#false\,matches$ is the number of candidate positions that were falsely identified as possible positions. Figure 8 contains the evaluation of our algorithm for pre-detection of the candidate positions. The more points are closer to the $[0;1]$ point, the more efficient and more precise the algorithm is. Since the data are not continuous and quite sparse we also show a Bezier curve approximation of the data points. Also notice that unlike in the case of local descriptor evaluation, the graphs end at $[1;0]$. This is caused by images where the algorithm could not find any potential code locations. The graphs show that our algorithm is quite robust and the size of the extra border areas has little effect on the results.

[1] Downloadable from `www.fit.vutbr.cz/research/groups/graph/download/qrcodes/`

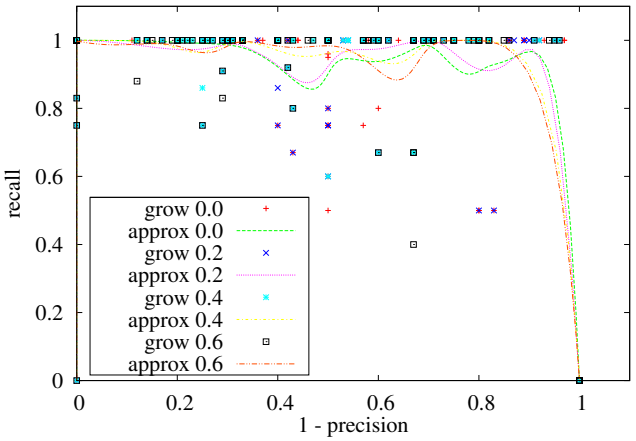


Figure 8: Evaluation of the candidate segments with varying extended size of the candidate positions. The *grow* parameters (0.2, 0.4, . . .) define relative enlargement of the detected rectangles.

type (count)	L	DT (100, 99, 95)	DC	ZB
plain (53)	93.7	90.6 (89.6, 95.8, 100.0)	88.7	90.6
rot (54)	98.2	88.9 (93.8, 97.9, 100.0)	85.1	98.2
pers (19)	100.0	94.7 (55.6, 61.1, 88.9)	63.2	57.9
rot+pers (50)	96.0	88.0 (77.3, 88.6, 95.5)	78.0	60.0
shadow (26)	92.3	69.2 (33.3, 66.7, 94.4)	46.1	46.1
blur (51)	80.4	52.9 (66.7, 74.1, 92.6)	41.2	67.9
overall (251)	92.8	80.5 (76.7, 86.1, 96.5)	70.1	74.8

Table 1: The results are shown for candidate position enlarged by 40 % of the corresponding segment size. The values in the three columns relate to the number of all QR codes in the category. L: successfully localized; DT: successfully detected (the brackets show the percentual correctness of the extracted matrices); DC: successfully decoded; ZB: success rate achieved by zbar library.

5.2 Detector evaluation

In this part we evaluated the overall precision of our method. We extended the method for evaluation proposed by Herout et al. [2012] to include information also on the localization performance. Since the QR code decoders are differently sensitive to the rotation, perspective deformation, blur and irregular illumination, the images are thus sorted into six categories:

plain the code is upright
rot the code is rotated
pers the code is upright but skewed by perspective
rot+pers general orientation of the code
blur general orientation of the blurred code
shadow general orientation of the code with irregular illumination

The algorithm is composed of localization, detection, and decoding of the codes. If the code is localized, the binary bitmap of the code's blocks (binary matrix) is extracted and compared to the ground-truth bitmap. Since the QR code is capable of error repair and a few percent of the code can be missed, correct detection is counted when at least 95 % (99 %, 100 %) of the code is correctly extracted. Finally, the QR code is decoded by a publicly available ZBar[2] library and compared to the original encoded string.

Table 1 shows the success rate for each step of the algorithm. All the values (except the ones in brackets) relate to the overall number of codes in the category. The first column shows the number of correctly localized barcode positions using the first part of the algorithm; the second column the percentage of the barcodes for which a binary matrix was successfully extracted. The brackets contain the information about the precision of the extracted binary maps compared to the ground truth data. The third column shows the percentage of the QR codes that were successfully decoded out of all the code instances in the dataset.

[2] `zbar.sourceforge.net` - we only reused the QR code decoding part from the already extracted matrix

For example for the rotated QR codes 98.1 % of the codes were correctly localized. For 88.9 % percent of all annotated QR codes, the binary matrix was successfully extracted. 93.8 % of these were identical to the annotated data, 97.9 % of them were over 99 % correct and all the rest was over 95 % accuracy. Lastly 85 % of the QR codes in this category were correctly decoded.

The results show that the blur caused by out-of-focus shots and movement are the most disturbing phenomena for 2D barcode detection by our algorithms. Another problematic case are 2D barcodes with shadows across them, since our detection algorithm does not take into account the much weaker gradients on shadowed areas. This could be solved by separating the candidate area into smaller parts and taking an equal number of best edge pixels from each part. Another disturbing area in the case of shadows is the penumbra, where the adaptive thresholding fails to determine the binary information of the field.

One other area where the matrix code is not extracted precisely are the perspectively distorted QR codes. This might be caused either by the non-planarity of surfaces and the small resolution of the $\mathcal{TS}$ space, since the maxima are not equally distributed along the vanishing line. To solve this issue, a re-fitting step could be added to detected QR code field centers similarly as was described by Parvu et al. [O. Parvu 2009].

We compared the performance and speed of our method with an open-source project for QR code detection: the zbar library. It uses linear scans with adaptive thresholding to localize FIPs in the QR codes. We chose this implementation because it is popular, frequently maintained, it uses a completely different and theoretically fast approach, and also because it supports detecting multiple QR codes in one image.

In Table 1 the ZB column shows the success rates for the zbar implementation for different QR code categories. Because zbar does not use edge extraction, it is more successful decoding blurred images. On the other hand it does not cope

well with perspectively distorted images in comparison to our method. The difference for detecting rotated QR codes might be caused by slight blur or small size of the codes, causing our algorithm to fail to determine the orientations precisely. This limitation might be solved with an extra sharpening step.

5.3 Speed of the Method

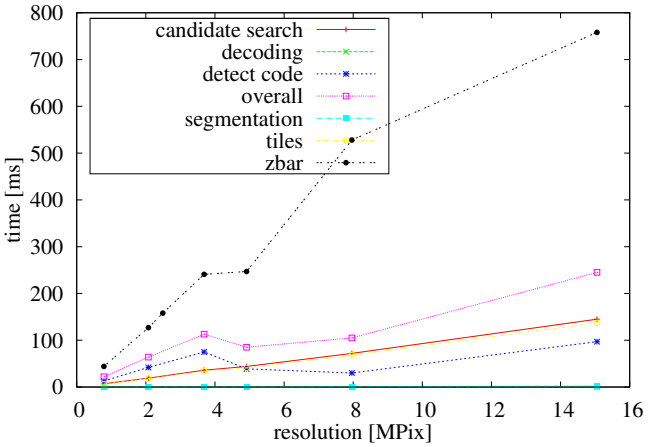


Figure 9: The required time for processing in ms. The graph also shows average required times for different parts of the algorithm: the overall time consists of the candidate search, code detection and binary matrix decoding; the code detection then from segmentation and the creation of the tile hierarchy.

Figure 9 shows the average required time for the algorithm. It does not contain the system overhead for loading and saving the images. The tests were carried out on a midrange Intel® Core™ i5-2410M, 2.3 GHz processor using a single core. The graphs show that there is a linear relation between the image resolution and candidate search. The overall time, however, varies more thanks to the different amount of detected candidate positions. The drop at 5 MPix resolution can be explained by the difference between professional cameras and cameras integrated into mobile phones. Since most mobile phones apply sharpening filters to the images, the number of edge gradients grow accordingly. This may cause a higher number of candidate positions, making the detection/recognition part of the algorithm slower.

The main purpose of using the grid of tiles is to reduce the number of positions to search for the QR codes in the whole image. As seen in Figure 9, the code detection itself takes for high-resolution images in average less than half of the required time of the whole algorithm. As mentioned earlier, to reduce speed and computational requirements we also approximated the HOG creation for the grid by subsampling the scanlines. This speedup is largely dependent on the distance of the scanlines. We used 10 pixel distance between consecutive scanlines. This means the algorithm had to process only approximately 1/5 of the image pixels plus the neighbors of the pixels where a gradient was detected to get the gradient direction.

From Figure 9 is also clear that the major slowing factors are the creation of the tile grid hierarchy and the number of candidate positions the detector is forced to process (see Figure 10). The grid creation part basically consists of the gradient detection and feature vector calculations for each tile. The segmentation is only done on the grid level and it does not introduce a large computational overhead. However, the quality of the segmentation and the classification of the segments has a direct effect on the number of candidate positions and consecutively on the speed of the detection part.

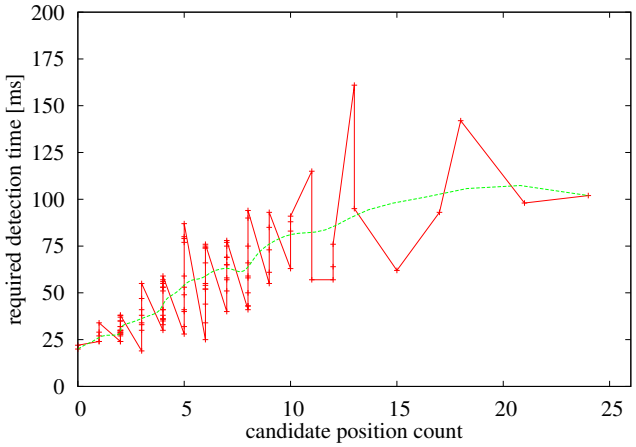


Figure 10: The required time for detecting QR codes in relation to the number of detected candidates in milliseconds for 1080p images.

We also tested the required time for resolution used by common HD cameras – 1080p. Figure 10 shows the results depending on the number of candidates positions that were found. The average time required to process one image was 54 ms and over 95 % of the images were processed under 100 ms. The outliers were caused mainly by extremely disturbing background texture and very high-resolution QR codes, where the number of candidates were relatively large.

Since we are able to detect for a large range of scales, the achieved 250 ms in average for 15 MPix images is sufficient for localization for a slow moving object equipped with a high-resolution camera. In order to get truly real-time speed we are planning on improving the segmentation and classification of the segments and speeding up the whole described algorithm using GPGPU solutions.

Figure 9 shows also the speed of the detection with the zbar library. Similar to the evaluation of our method, these results do not include the overhead for loading and saving images. The results show that the zbar library is approximately 4 times slower for high resolution images with a comparable detection rate.

6 Conclusions

This paper introduces an algorithm for predetection of QR codes (and similar matrix codes, such as Aztec, DataMatrix, etc.). The algorithm is capable of detecting locations with probable occurrences of matrix codes in a high-resolution image. These locations can be further processed by an algorithm for detection and recognition of QR codes; the paper summarizes our recent algorithm based

on Hough transform. The predetection algorithm and the defection/recognition one can share some computed information, further reducing the computational cost.

The algorithm was evaluated on a dataset of high-resolution images with one or more QR codes present in the scene. The codes are viewed from different point and at different scales; the dataset thus well represents the real-life scenes. Our algorithm successfully localized 95 % of all QR codes. The required run-time was under 250 ms in average for the tested images (up to 15 MPix large).

The algorithm is fairly efficient and it notably exceeds the results published recently (e.g. [Belussi and Hirata 2011]). To evaluate the performance of the algorithm, we collected and annotated a dataset of real-life images, suitable for comparison of algorithms. Such a public dataset was missing; making the dataset available is one of the contributions of this paper.

The algorithm is very suitable for parallelization in an SIMD environment and it should be fairly efficient when executed on recent GPUs. In near future, we intend to produce an OpenCL implementation of the algorithm. We are expecting the GPU implementation of this algorithm to be runnable in real time even on very high resolution images.

Acknowledgements

This research was supported by the research project CEZMSMT, MSM0021630528, by the CEZMSMT project IT4I - CZ 1.05/1.1.00/02.0070, and by the BUT grant FIT-S-11-2.

References

Alfthan, J. 2008. *Robust Detection of Two-Dimensional Barcodes in Blurry Images.* Master's thesis, KTH Computer Science and Communication, Stockholm, SE.

Arnould, S., Awcock, G., and Thomas, R. 1999. Remote bar-code localisation using mathematical morphology. In *Seventh International Conference on Image Processing and Its Applications*, vol. 2, 642 –646 vol.2.

Belussi, L., and Hirata, N. 2011. Fast QR code detection in arbitrarily acquired images. In *Conference on Graphics, Patterns and Images, SIBGRAPI 2011*, 281 –288.

Dalal, N., and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *Proceedings of CVPR 2005*, vol. 1, 886–893.

Denso Wave Inc., 2010. QR code information homepage. `http://www.qrcode.com/index-e.html`.

Dubská, M., Herout, A., and Havel, J. 2011. PClines – line detection using parallel coordinates. In *Proceedings of CVPR 2011*.

Duda, R. O., and Hart, P. E. 1972. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM 15*, 1, 11–15.

Haindl, M., and Mikes, S. 2005. Colour texture segmentation using modelling approach. In *ICAPR (2)*, 484–491.

Herout, A., Dubská, M., and Havel, J., 2012. Real-time precise detection of regular grids and matrix codes. Submitted to ECCV2012.

Hu, H., Xu, W., and Huang, Q. 2009. A 2D barcode extraction method based on texture direction analysis. In *Fifth International Conference on Image and Graphics, ICIG '09.*, 759 –762.

Jain, A., and Chen, Y. 1993. Bar code localization using texture analysis. In *Second International Conference on Document Analysis and Recognition*, 41 –44.

Liu, Y., Yang, J., and Liu, M. 2008. Recognition of QR code with mobile phones. In *Chinese Control and Decision Conference, CCDC 2008*, 203 –206.

Mikolajczyk, K., and Schmid, C. 2005. A performance evaluation of local descriptors. *IEEE Tr. on Pattern Analysis and Machine Intelligence 27*, 10, 1615 –1630.

Muniz, R., Junco, L., and Otero, A. 1999. A robust software barcode reader using the hough transform. In *International Conference on Information Intelligence and Systems*, 313 –319.

O. Parvu, A. B. 2009. A method for fast detection and decoding of specific 2d barcodes. In *17th Telecommunications forum TELFOR 2009*, 1137 – 1140.

Ohbuchi, E., Hanaizumi, H., and Hock, L. 2004. Barcode readers using the camera device in mobile phones. In *Int. Conference on Cyberworlds (CW'04)*, 260 – 265.

Ojala, T., and Pietikäinen, M. 1999. Unsupervised texture segmentation using feature distributions. *Pattern Recognition 32*, 3, 477–486.

Ouaviani, E., Pavan, A., Bottazzi, M., Brunelli, E., Caselli, F., and Guerrero, M. 1999. A common image processing framework for 2D barcode reading. In *International Conference on Image Processing and Its Applications*, vol. 2, 652–655.

Parikh, D., and Jancke, G. 2008. Localization and segmentation of a 2D high capacity color barcode. In *IEEE Workshop on Apps. of Computer Vision (WACV)*, 1–6.

Rosenfeld, A., and Pfaltz, J. L. 1966. Sequential operations in digital picture processing. *J. ACM 13* (October), 471–494.

Sun, A., Sun, Y., and Liu, C. 2007. The QR-code reorganization in illegible snapshots taken by mobile phones. In *International Conference on Computational Science and its Applications*, 532 –538.

Wang, H., and Zou, Y. 2006. Camera readable 2D bar codes design and decoding for mobile phones. *IEEE International Conference on Image Processing*, 469–472.

Weldon, T. P., Higgins, W. E., and Dunn, D. F. 1996. Gabor filter design for multiple texture segmentation. *Optical Engineering 35*, 2852–2863.

Xue, Y., Tian, G., Li, R., and Jiang, H. 2010. A new object search and recognition method based on artificial object mark in complex indoor environment. In *8th World Congress on Intelligent Control and Automation (WCICA)*, 6648 –6653.