QR code processing

Advanced Image Processing, TNM034

Linköping University

Louise Carlström	louca859
Karolin Jonsson	karjo132
Linnéa Mellblom	linme882
Linnéa Nåbo	linna887

December 11, 2014

Contents

1 Introduction

A QR code is a type of two dimensional barcode used for storing information. There are 4 standardized coding modes; numeric, alphanumeric, 8-bit bytes (binary) and kanji [1]. In this report only 8-bit binary QR codes will be considered.

By decoding the QR code byte by byte and translating it into ASCII code, a message can be obtained. The code can be read using a barcode scanner and the decoded message usually contains an informative text about the object to which it is related to, or a link to a website etc.

The QR code is made up of white and black squares called modules. There are currently 40 versions of QR codes and each version has a different number of modules. This report will only focus on QR version 6 which consists of 41 x 41 modules.

The QR code contains a number of finding patterns (FIP) which are used to locate and recognize the code. It also contains alignment patterns (AP) which are used in geometric calibration to reconstruct a perfectly shaped code. The FIPs and the AP for QR version 6 can be seen in Figure 1.

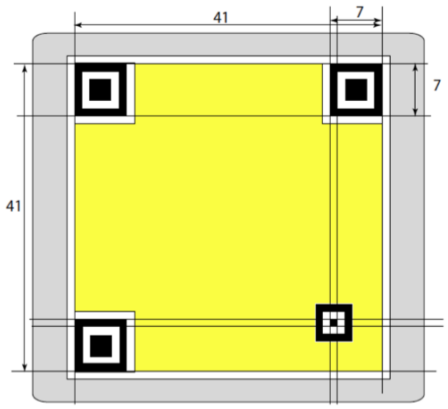

Figure 1: *The QR code (version 6) layout. 41x41 modules. Three FIP and one AP.*

2 Method

This project has been carried out using MATLAB, it's built-in functions and the image processing toolbox. Ideas and algorithms has been inspired by previous knowledge gained in other courses and collected from scientific reports and documents about QR standards and pattern recognition during the initial researching phase.

2.1 Thresholding

In order to find and read the QR code in an image some alternations of the original image are required. The most important aspects is that the image is binary (all pixels are of value 1 or 0) and that the binary image does not contain too much noise. A noisy image contains more internal subpatterns, making it harder to find the finder patterns of the QR code.

Before processing the QR code it needs to be converted into a binary image with the help of a threshold. This threshold is found by using the built in MATLAB function *graythresh* [2]. It is a global thresholding function using Otsu's method for automatic clustering-based thresholding. The algorithm uses bimodal histograms to classify the pixels in an image into two classes or clusters, an optimal threshold is thereafter calculated to separate the two pixel clusters. This threshold is then used together with MATLAB's built in function *im2bw* [3] to create a binary image which in turn can be used to identify the QR code.

2.1.1 Alternative method for thresholding

The global thresholding approach works well for evenly contrasted images. In images which are for example unevenly lit, it gives rise to errors. To be able to manage images with large contrast differences, an adaptive thresholding approach using locally calculated thresholding values can be used.

Firstly a certain amount of padding needs to be added to the image in order to be able to threshold it. The image must be of a size that makes it possible to fit an exact amount of thresholds into the image. No padding can lead to incorrect thresholding along the edges of the image.

The image is then divided into a number of sections and *graythresh* is used within each section to calculate several local threshold values. The number of thresholds to be used can possibly be calculated based on the number of unique levels of intensity in the image in combination with the size of the QR code relative to the entire image.

The images which do not have a problem with a large range of contrasts or where the QR code take up most of the image will however still be using one single global threshold.

After the entire image has been transformed into a binary image, the padding is removed.

The reason some images did not work with the adaptive thresholding is that too many thresh-

olds were used and this resulted in some sections where black sections were thresholded into black and white anyway.

To solve this an allowed margin of error is added. Before thresholding the difference between the maximum and the minimum value within each section is calculated. If this difference is small, the section is chosen to be either entirely black or entirely white depending on if the average of the difference is above or below the global threshold value.

The adaptive thresholding approach unfortunately did not work for all test images, therefore, a global threshold is used in the final version.

2.2 Pattern Recognition

In order to identify the QR code in the image, some patterns are searched for: three finder patterns (FIP) and the alignment pattern (AP). When these patterns are located, the image can be transformed so that the QR code can be decoded correctly.

In all of these finder and alignment patterns a specific subpattern can be found [8]. The pattern consists of the relationship and ratio between black and white modules in the image. The relation is the same regardless of how the image is approached and hence the image can be scanned both horizontally, vertically and diagonally. This means that the subpattern can be found independent of the rotation of the QR code. Where the subpattern is found in more than one direction there is a possibility that the intersection point is a finder or alignment pattern. The Figure 2 demonstrates the subpatterns of a FIP [5].

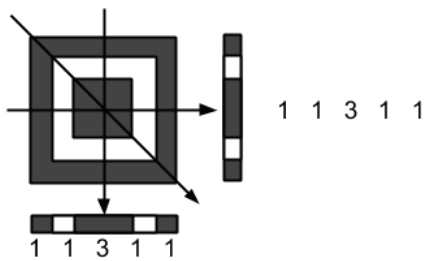

Figure 2: *Can search different directions and still find the middle of the pattern .*

2.2.1 Locate Finder Pattern, FIP

The FIPs have a specific relationship that can be searched for: modules of 1:1:3:1:1, as seen in Figure 2. A module is made up of contiguous pixels in black or white and in order to search for these patterns, the image is scanned. This scan is resulting in an output similar to Figure 3.

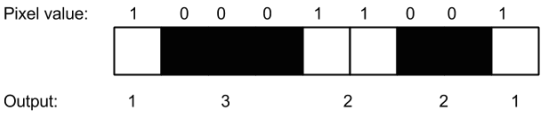

Figure 3: *A modules size. Contiguous pixels in row .*

The algorithm search row by row for the specific finder pattern. At each row five adjacent values are selected and a check is performed to determine if these correspond to the desired pattern of 1:1:3:1:1. A certain error is allowed and the tolerance is 0.3. This value has been derived through empirical studies. If the desired pattern has been located, the algorithm also checks if the first pixel value is black. If that is true, the center column of that pattern is searched top down using the same approach. If the 1:1:3:1:1 pattern is found in the column search and is located so that it corresponds to the row where the first 1:1:3:1:1 pattern is found, a FIP candidate has been found and the center pixel value is saved. The algorithm then continues to search the image row by row.

After this step, the algorithm should have been able to find several FIPs. In order to sort out the three candidates of the potential finder patterns the clustering algorithm k-means is used. K-means clustering is an iterative partitioning method. The function *kmeans* in MATLAB [9], by default uses the k-means++ algorithm for initializing the centroids of the clusters and the distance is calculated using the squared Euclidian metric. Pixel values who are located close to each other and far from others are grouped together in a cluster, each pixel value belongs to the cluster with the nearest mean. The algorithm is iteratively minimizing the distance from each point in the cluster to its cluster centroid and pixels are moved between clusters until the sum has been minimized, resulting in compact sets of clusters which are as separated from each other as possible [10]. This is resulting in three distinct clusters in which the pixel closest to the mean value of each cluster is chosen as a FIP.

The algorithm should now have found three FIPs. In order to be able to do the geometric calibration, the algorithm should determine which finder pattern is the top left (A), bottom left (B) and top right (C). The top left FIP is located first. This is done by calculating the distance between the three FIP positions. The relationship between

the locations of the FIPs can be seen as a triangle. The largest calculated distance will be the hypotenuse of this triangle, therefore it is possible to classify the FIP which is not associated with the hypotenuse as the top left FIP.

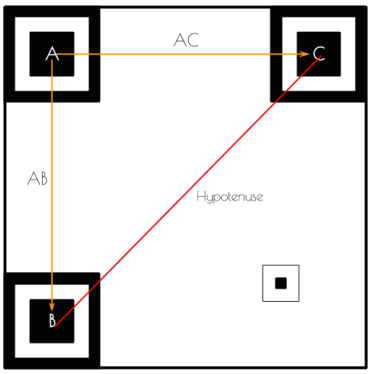

Figure 4: *Determine locations of the FIPs* .

In order to determine which of the remaining FIPs are the top right and bottom left, the perp product is used. Two vectors are calculated between the three FIP positions, AB and AC (see Figure 4). In this state, the relation between the vectors is unknown. From the perp product between these vectors (Figure 5), the angle between the vectors can be derived. The top right and bottom left FIP can then be determined based on the angle between the two vectors AB and AC, as follows:

$$\begin{aligned} \vec{AB} \perp \vec{AC} = 0 \Leftrightarrow \\ \text{AB and AC are collinear. i.e.} |\theta| = 0 or 180°. \\ \vec{AB} \perp \vec{AC} > 0 \Leftrightarrow \\ \text{AC is left of AB i.e.} 0 < \theta < 180°. \\ \vec{AB} \perp \vec{AC} < 0 \Leftrightarrow \\ \text{AC is right of AB. i.e.} 0 > \theta > -180°. \end{aligned} \quad (1)$$

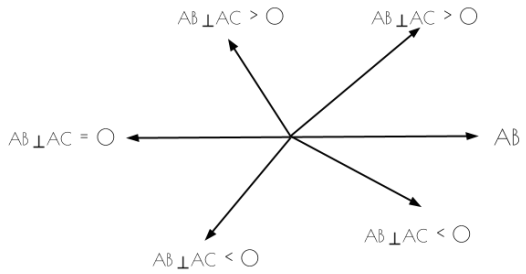

Figure 5: *To identify the location of the top right FIP and lower left FIP* .

In short, the algorithm used to locate the finder pattern is as follows:

1. Search for the 1:1:3:1:1 pattern in each row and if found:
2. Search for the 1:1:3:1:1 pattern in the middle column of the found pattern
   (a) If a pattern is found in the column and its center is located on the same row as in 1: store as a FIP candidate
3. Thin out the FIP candidates down to the three real FIP locations.
4. Locate which FIP is the top left, bottom left and top right

2.2.2 Locate Alignment Pattern, AP

In order to locate the AP, the relationship 1:1:1 is searched for. This is done using the same approach as to find the FIPs, but with the difference that the first pixel/module should be white instead of black. In order to narrow down the search field of the AP, a position that should theoretically be close to the AP is calculated. This can be done since the standards of the QR code says that the AP should lie in a specific relation to the FIPs (can be seen in Figure 1 in the introduction). The FIP positions are known at this stage and an approximated position of the AP can be calculated as:

$$cellWidth = \frac{max(|AC|, |AB|)}{34} \quad (2)$$

$$AC_n = \frac{AC}{|AC|} \quad (3)$$

$$AB_n = \frac{AB}{|AB|} \quad (4)$$

$$(5)$$

$$p = AC_n * 31.5 * cellWidth + AB_n * 31.5 * cellWidth + A \quad (6)$$

The value 31.5 in equation 6 corresponds to 31.5 modules and is used since the distance between A:s position and the middle of the first black module in C (see Figure 4) will be 31.5 modules. As can be seen in Figure 6, the centre of the AP will be located in the same column as the first black module in FIP C and on the same row as the beginning of FIP B, if perfect conditions prevail.

The algorithm then searches for the relationship 1:1:1 in the image and saves the AP candidates. The AP candidate nearest the calculated point p is chosen.

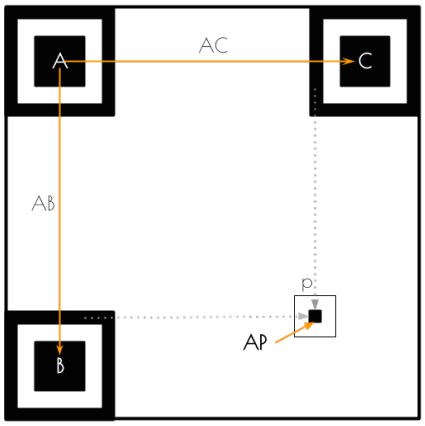

Figure 6: *How to find the AP .*

2.2.3 Alternative methods for FIP and AP detection

Alternative methods for localizing the QR code would be to search for contours in the image, look at the histogram or the frequency information [16].

Instead of searching for the 1:1:3:1:1 pattern to find the FIPs it would be possible to use the intensity histogram to discover features. An intensity histogram displays the contribution and distribution of each level of intensity in the image. The advantages of intensity histograms are that they are scale, position and rotation invariant. But since histograms do not contain any spatial information it may be hard to discriminate features only based on the histogram [13].

2.3 Geometric Calibration

There are several methods available to transform a distorted QR code into a readable square shape. The basic requirements depend on how distorted the image is allowed to be and what hardware is available. For a geometrically distorted image, transform matrices can be used, such as rotation, scaling, translation and skewing matrices. If the QR code has other distortions such as if the image is curved, simply using a transformation matrix will not suffice. In this project the possibility of a curved image is excluded, and the report focuses on geometric distortions.

Basic transformation matrices such as rotation or scaling matrices can be used to straighten the image. However, this requires a lot of knowledge about the original image and calculations of rotational angles etc. In many cases, the QR code is not only rotated but can also diverge from being a quadratic image in other ways, it can for example be both sheared and scaled. In those cases, a series of matrix multiplications would be required in order to reach the desired result.

QR codes are primarily used with mobile devices, which limits the hardware capacity for reading the codes. Therefore, a sequence of matrix multiplications could easily become too heavy for the device, causing it to crash. In order to reduce the amount of matrix multiplications a perspective transformation [14] can be computed. Points in the original image and corresponding points in a reference image are found, the points are chosen in suitable positions in the QR code. From the relation between the two sets of points the matrix can be derived. The more points used to calculate the transformation matrix, the better the outcome of the transformation will be.

In the case of QR codes of version 6, there are three finder patterns and one alignment pattern in the image, which means four points can be used. Despite the low number of points the perspective matrix can be suitable enough in a wide range of images. To find the straight image I_s it can be seen as the distorted image I_d with every point of the image multiplied with a transformation matrix X as follows 7

$$I_s = I_d * X \tag{7}$$

In order to derive the transformation matrix, the set of points in each image are used in the same equation P_s and P_d, the inverted matrix of the distorted points is used 8.

$$X = P_d^{-1} * P_s \tag{8}$$

Once the transformation matrix is known, all pixels of the straight image can be calculated. In this case, the transformation matrix was derived through the MATLAB function *fitgeotrans* [11] function and the transformation done by using the MATLAB linear *imwarp* function [12]. Initially a custom function for these calculations was created, in which the pixels were individually multiplied with the transformation matrix. This was sufficient for most images, but failed for images with perspective distortions. When the same points were used together with the MATLAB built in functions the result showed a remarkable improvement.

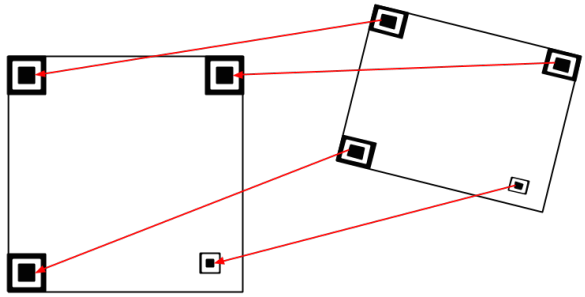

Figure 7: *The straight and the distorted images and the four points used to calculate the transformation matrix .*

To further stabilize the reading, a combination of the two approaches can be used. Putting aside the need for hardware efficiency opens a possibility to combine transformation executions.

In this case, though, a perspective transformation was enough for all tested images, assuming the FIPs and AP were found in the image. This is considered sufficient within the scope of this report since there is no need to create a heavier program when in leads to no visible improvement.

2.4 Decoding

When decoding the QR code, a reference image (Figure 8) is used to mask the FIPs and the AP so that these are not incorporated in the message.The QR code is scaled down to 41 x 41 pixels and is read column by column from left to right. The image is converted into vector form and the masked pixels are removed. The vector now only contains the actual data that will be read. The image is then converted into a binary string and divided into 8 bit strings, which are translated into a decimal number and translated into a character according to the ASCII table.

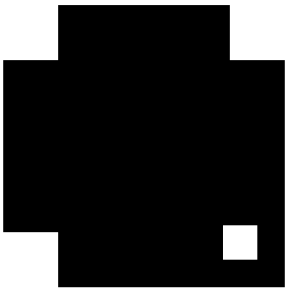

Figure 8: *A mask to cover the FIP and AP .*

3 Result and discussion

The resulting program operates on a set of images of various kinds. The images in the test set are distorted in different ways. Some are rotated, some compressed and some have noise or irregular lighting. There are also different combinations of these distortions. The test sets are therefore suitable for testing the program with respect to the specified requirements.

One of the problems encountered occurred when scanning the image for potential FIPs in rows and columns separately. This initial approach resulted in sets of vectors of 1:1:3:1:1 patterns for rows and columns and if the exact same pixel value was found in both column and row vector the common pixel represented the center of a potential FIP. This was problematic since the same pixel was not always present in both the column and the row vector, resulting in some FIPs being overlooked. One idea of how to solve this was to calculate the start and end point of the FIP in both the column and row vector and calculate two perpendicular lines (one for the row and one for the column) in between these points and let the intersection between them be the center of the FIP. This approach is similar to the one described in the ISO/IEC 18004 standard [15]. Since this would have meant rewriting much of the code the problem was solved by introducing a tolerance value. This made it possible to search also the adjacent pixels and find FIPs even though the column and row vectors did not contain exactly the same pixel. This in turn lead to the gathering of several potential center of FIPs in the same area. To solve this the k-means clustering algorithm was applied (see section 2.2.1), resulting in one FIP per area.

The transformation algorithm is sufficient for most geometric distortions and can handle big rotation angles and non quadratic images, like the test images of this project. However, the transformation matrix might have some weaknesses if the distortions are more extreme. A possible action to avoid this could be to use a different method for decoding the image. Currently, the transformed and cropped image is resized to an image of the size 41 x 41 pixels, which leaves little room for remaining distortions in the transformed image. Another approach could be to find other complementary transformation executions or more points by using for example line detection.

One weakness in the application is reading images with extreme lightning differences throughout the image. In some cases, adaptive thresholding is necessary. There are different ways to accomplish adaptive thresholding. The approach used in

this case worked perfectly on some images, but the problem was the variety of image sizes in relation to the size of the QR code in the image. With the wrong constants in the algorithm the binarization in some cases lead to increased amount of noise in the image, creating a more difficult environment for locating the finder patterns and the alignment pattern in the image. To a wide extent, though, global thresholding could be used, and this was applied in the successful tests.

Finally, decoding one image takes an average of approximately 2.3 seconds, depending of the size and complexity of the image.

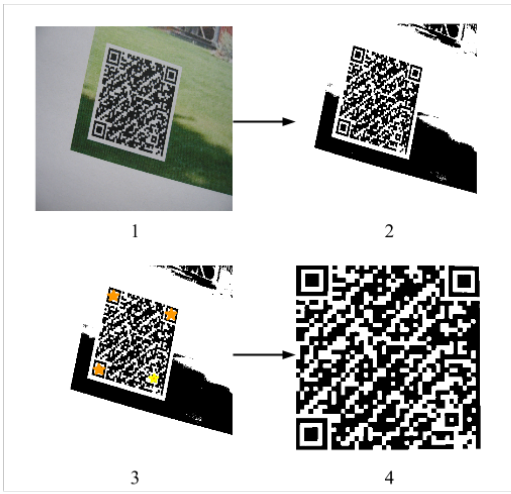

Figure 9: *The original image is recalculated into a binary image (1 - 2), thereafter the FIPs and the AP are found and the matrix transformation is performed (3 - 4) .*

4 Conclusion

In this project a MATLAB application for detecting, reading and decoding QR codes of version 6 has been developed. The resulting application is fairly fast and overall it provides a fairly accurate result.

QR code detection relying on the alternation between black and white pixels to find FIPs is somewhat risky since the regularity can be disrupted in low resolution images. During the course of the project problems were encountered regarding FIP detection in noisy areas, since multiple false FIPs are found. One approach to make solve this, and to make the algorithm more stable in noisy pictures, is to store the module size of the FIPs found and compare these to determine which ones are the true FIPs.

The program created is considered to solve the problem within the scope of the project.

References

[1] Standards of QR codes. `http://www.qrcode.com/en/about/standards.html` December 11, 2014

[2] MATLAB-function graytresh `http://se.mathworks.com/help/images/ref/graythresh.html`

[3] MATLAB-function im2bw `http://se.mathworks.com/help/images/ref/im2bw.html`

[4] Different versions of QR codes. `http://www.qrcode.com/en/about/version.html)` December 11, 2014

[5] QR code, Section 3 `https://foxdesignsstudio.com/uploads/pdf/Three_QR_Code.pdf` December 11, 2014

[6] Yue Liu, Mingjun Liu, *Automatic Recognition Algorithm of Quick Response Code Based on Embedded System.* Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (2006): (ISDA'06) 0-7695-2528-8/06

[7] Chung-Hua Chu, De-Nian Yang, Ya-Lan Pan, Ming-Syan Chen, *Stabilization and extraction of 2D barcodes for camera phones.* Multimedia Systems (2011): 17:113–133 DOI 10.1007/s00530-010-0206-9

[8] Luiz F. F. Belussi1, Nina S. T. Hirata1 *Fast Component-Based QR Code Detection in Arbitrarily Acquired Images* Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, Rua do Matão, 1010, 05508-090 São Paulo, Brazil

[9] K-means algorithm `http://se.mathworks.com/help/stats/kmeans.html#bueftl4-1` December 11, 2014

[10] K-means clustering `http://se.mathworks.com/help/stats/k-means-clustering.html` December 11, 2014

[11] Fitgeotrans function `http://se.mathworks.com/help/images/ref/fitgeotrans.html` December 11, 2014

[12] Imwarp function `http://se.mathworks.com/help/images/ref/imwarp.html` December 11, 2014

[13] FIP detection `http://www.nada.kth.se/utbildning/grukth/exjobb/`

rapportlistor/2008/rapporter08/alfthan_jonas_08033.pdf) December 11, 2014

[14] George Baravdish, *Linjär algebra, TNA002* Linköping University, 2011.

[15] http://raidenii.net/files/datasheets/misc/qr_code.pdf , page 61/67 December 11, 2014

[16] Luiz F. F. Belussi, Nina S. T. Hirata *QR Code Detection in Arbitrarily Acquired Images* http://www.ucsp.edu.pe/sibgrapi2013/eproceedings/wtd/114652.pdf