Image analysis methods for localization of visual codes

Ph.D. thesis

Péter Bodnár
Advisor: László G. Nyúl, Ph.D.

PhD School in Computer Science
Department of Image Processing and Computer Graphics
Faculty of Science and Informatics
University of Szeged

Szeged, 2015.

Contents

Chapter 1

Introduction

The application of computer-readable visual codes has become common in our everyday lives. Their importance has increased not only within industrial environment, but in private use as well [26]. Computer-readable visual codes are universal, and their production is less expensive than using other technologies, such as radio frequency identification (RFID) [24]. They provide simple and reliable identification of items at postal services, point of sale (PoS) terminals, and warehouses of various products. Moreover, they can be used as tickets [66], for inventory control and for tracking purposes as well. The availability of codes using desktop printers, and penetration of automatic checkout systems at supermarkets [40] also greatly contributed to the popularity of these codes.

The history of visual codes began with the application of barcodes. These one-dimensional codes represent embedded data using parallel line segments (bars) of various widths, aiming easy identification. Later, this idea evolved into using two-dimensional layouts of discs, hexagons, or squares. Despite their structure, 2D codes are also often referred as barcodes.

The reading process of visual codes consists of two steps, namely, localization and data decoding. In the first step, the presence of the barcode, and the location of the barcode to the sensor has to be recognized. In the early age of barcodes, localization was manual. A data terminal or a portable barcode scanner was positioned and oriented against the item having the code object, in order to recognize it. Barcode reading on recent smartphones is still at this stage.

Since the past couple of years, image acquisition techniques and computer hardware have also improved significantly, automatic reading of visual codes became available [84], however, it raised the localization problem to the next level of difficulty. Visual codes have to be reliably found without human assistance, based on the

sensor data. Each application has its properties, like, the expected type of barcode, type of the sensor, and constraints of size, orientation, and number of codes present within the observed area. After the successful localization of the code, the decoding step follows, which means retrieval of the embedded data by the software. While the localization step became more complex due to the expectation of automatism, the reliability of decoding can be contributed to the availability of more accurate sensory data and additional computational capacity.

As applications impose special problems, there is a continuous need for solutions with improved effectiveness. There are several methods for barcode localization that are characterized by the processing techniques they use, accuracy and speed. However, state of the art algorithms, while giving efficient methods to the decoding step, still lack universal solutions for localization. This implies the need of research in that step of the reading process. The primary objective of this thesis is to examine and improve existing algorithms for barcode localization, and to design new ones.

Recent trends in computational capacity also allow using machine learning approaches and classifiers based on more complex features [3, 70] than the ones used at barcode scanner devices.

1.1 Barcode formats

Barcodes have been used since the 1960s, when General Telephone and Electronics invented the Automatic Car Identification method [74]. A barcode is a code that originally consisted of colored lines, placed on both sides of the railroad trucks. This code contained the identification number, or some owner information, and was read by scanners at the entrance gates. Ten years later, barcodes became wide-spread at supermarkets and other places for Automatic Identification and Data Capture (AIDC) [65]. Until the introduction of Radio Frequency Identification (RFID) [24], barcode technology dominated in all tasks of identification and tracking.

1.1.1 One-dimensional barcodes

In the early years of the barcode era, blood banks [50] and libraries used the Codabar barcode type (Fig. 1.1(a)). It could be printed on a typewriter, which led to its popularity. Moreover, it also contained a check digit, which was rarely used in early schemes.

The National Association of Food Chains raised the idea of automated checkout systems at supermarkets [51], which resulted in one of the most popular barcode

(a) Codabar (b) UPC-A (c) UPC-E

(d) Code 39 (e) Code 128 (f) EAN-13

Figure 1.1: 1D barcode formats.

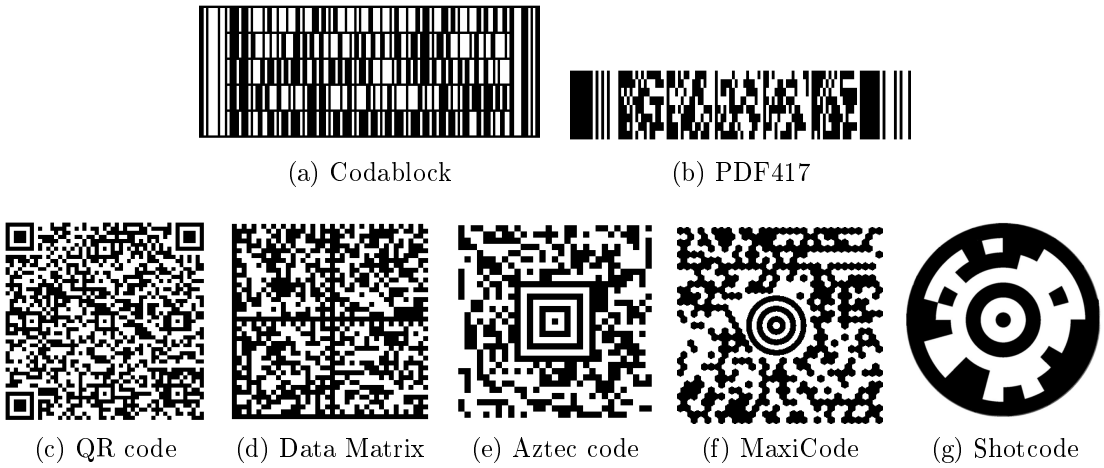
patents, the Universal Product Code (UPC). This code type has two variants. The UPC-A variant (Fig. 1.1(b)) can embed 12 digits, which is typically a unique ID for items at the point of sale. The UPC-E variant (Fig. 1.1(c)) is used on smaller products, where the UPC-A code can not fit, and it uses only 6 digits.

Other commonly used code types, such as Code 39 (Fig. 1.1(d)) and Code 128 (Fig. 1.1(e)) are also international standards (ISO/IEC 16388 and ISO/IEC 15417). The US Department of Defense used Code 39 for their products. It defines a total of 43 symbols, uppercase alphabet, digits and some special characters (-, ., $, /, +, % and space). Its data density is lower compared to Code 128, however, both are widely used. Code 128 can encode all 128 ASCII characters, which rendered it a general purpose format.

The GS1-approved EAN code family was first being used on books, but also became a general purpose barcode type with EAN-8 and EAN-13 patents (ISO/IEC 15420) [21]. EAN-13 (Fig. 1.1(f)) is a superset of the 12-digit UPC-A format, including a check digit.

1.1.2 Two-dimensional barcodes

The next generation of barcodes represent data along two dimensions. 2D codes have similar use like their 1D predecessors, but they have higher data density and



(a) Codablock (b) PDF417

(c) QR code (d) Data Matrix (e) Aztec code (f) MaxiCode (g) Shotcode

Figure 1.2: 2D barcode formats.

more complex layouts. Also, there are stacked 1D codes, like Codablock [43] and PDF417 [43], that use 1D barcode technology, but embed information along both dimensions.

Codablock (Fig. 1.2(a)) is developed by ICS International in the late 1980s. Its representation is similar to line wrapping in text editors. Characters are still encoded using lines of various widths, but the data is wrapped in up to 22 lines. Each line can contain up to 61 characters, thus making a Codablock code capable of encoding 1340 characters. PDF417 has similar layout (Fig. 1.2(b)). Each pattern consists of 4 bars, and each pattern is 17 units long. Besides Aztec code, it is still a popular format for airline boarding passes [56], and for the US Postal Service.

Most 2D barcodes contain helper patterns for easier localization and decoding, however each type has a slighly different approach. Most of them contain one or more bulls-eye patterns (localization pattern, finder pattern, FIP), data density patterns, checksums or Error Correcting Codes (ECC). The most popular ones are QR code, Data Matrix, Aztec Code, MaxiCode and Shotcode [43].

Quick Response (QR) code (Fig. 1.2(c)) is designed by the Japanese automotive industry (ISO/IEC 18004:2000). It has many variants with respect to size, level of error correction, and available encoded data type. They can encode numeric, alphanumeric, binary information and kanji, and they use Reed-Solomon error correction. The format was originally designed for industrial use, but it became popular in almost every task of identification and tracking. Typically, it can embed URLs that smartphones can open, then they navigate to the website of the corresponding item the code was scanned from.

Data Matrix (ISO/IEC 16022:200) (Fig. 1.2(d)) can encode text or numeric data,

from a few bytes to 1556 bytes, and also use ECC. Its main difference from other 2D codes is that both its finder and data density patterns are on the perimeter of the code object, and it contains no finder patterns. Its use ranges from grocery product marketing to industrial item marking [28].

Aztec Code (ISO/IEC 24778:2008) (Fig. 1.2(e)) has given its name from the bulls-eye pattern in its center that resembles to an aztec pyramid. Its significance is in that it uses less space than its 2D siblings since it does not require blank space (quiet zone) around the code object. It is mostly used for railway tickets and boarding passes [56].

MaxiCode (ISO/IEC 16023) (Fig. 1.2(f)) uses dots, arranged in a hexagonal grid, and it has a round bulls-eye pattern in the middle. It is used on invoices, item tracking and customer services.

ShotCode (originally SpotCode) (Fig. 1.2(g)) is created in Cambridge University as a low-cost solution to track locations. It has a fully circular layout for the embedded data, with a bulls-eye pattern in the middle.

1.2 The barcode reading process

We have to take two steps to regain the data embedded in the visual code. First, we locate the barcode, recognize its size, orientation, and usually apply transformations, like noise reduction, sharpening, normalization, and correction of distortions. After this step, the processed image piece containing the code is passed to the detector that looks up the pattern for valid character data. When proper localization and preprocessing is applied, then this step is relatively straightforward since characters are easily recognizable thanks to the maximized hamming-distances between entities. In addition, most barcode standards also provide redundant information for error correction purposes. Localization step has more difficulties due to the variety of code types, cameras and setups.

Barcode localization methods have two objectives, *speed* and *accuracy*. For industrial environment, accuracy is crucial since undetected (missed) codes may lead to loss of profit. Processing speed is a secondary desired property of the detectors. On smartphones, accuracy is not so critical, since the device interacts with the user and re-shooting is easily done, nevertheless, fast (and reasonably accurate) barcode detection is desirable.

There are various approaches for localization, from using the simplest features to machine learning and complex image manipulation methods.

1.3 Thesis outline and contributions

In Chapter 2, I study algorithms using global information, extend the oldest method of localization, namely, the scan-line approach, and design a new algorithm using Hough transformation. Lastly, I evaluate methods involving mathematical morphology, and give a variant of that algorithm group, that has high recall rate. This latter attribute makes this algorithm suitable especially for industrial setups.

Chapter 3 focuses on methods based on image tessellation, and simple features that can be computed locally. Those algorithms are the best regarding running time and memory cost, which makes them ideal for smartphone applications. The scan-line approach is revisited, in sections 3.1.1 and 3.1.2, I give two new modifications for the basic scan-line algorithm, for making local measurements. In the rest of Chapter 3, I introduce new local approaches and evaluate them individually and in ensemble. Algorithms of that chapter can be considered fast because of the possibility of parallelization.

In Chapter 4, I introduce the neural networks for the localization task. It shall be highlighted that both conventional and deep networks are tested in this research. An interesting contribution to the neural network training is the exploitation of their learning ability directly in the frequency domain, thus taking advantage of the JPEG image format. Moreover, neural networks can also evaluate input vectors in parallel.

In Chapter 5, I study and improve the performance of cascade classifiers for the localization task. A cascade classifier was first introduced by Belussi et al. [3]. I propose an improvement on the classifier training target, and evaluate classifiers based on new features.

As the last chapter (Chapter 6), I give an alternative approach for the localization task in the form of Fuzzy Inference Systems (FIS). Features for barcode localization are easier to be expressed using vague terms or value ranges. Terms of the Fuzzy set theory and rules of the Fuzzy Inference Systems give a convenient tool for the task in a more relaxed manner, instead of having to define and optimize parameters and constants.

Additionally, it shall be noted that both cascade classifiers and FIS can be considered as robust, fast algorithms, based on more complex features, yet they can be executed in real time.

Chapter 2

Algorithms using global image data

In this section, methods are presented that use features based on global information, which means all sensor data is available for the algorithm at the same time. The very first idea was the scan-line analysis, that came from the early age of computer vision, where mathematical morphology [62] was too time-consuming to perform on high resolutions. However, the spreading of embedded systems renewed the interest of these fast algorithms, and gave motivation to improve existing algorithms and develop new ones. Moreover, these algorithms can be designed, verified and fine-tuned more easily than machine learning approaches.

There is a wide selection of papers on scan-line analysis [1, 27, 72, 73, 76], having the same simple idea. 1D intensity profiles are acquired along sweeping lines of the sensor area, and further processed by the decoder. Although scan-line based solutions are fast, they have low tolerance to noise and smoothing. I examined the scan-line method and evaluated possible modifications.

Methods involving mathematical morphology use operations derived from erosion and dilation [36,38,47]. This group of algorithms has the common properties of larger computational cost and higher level of robustness. I also proposed an algorithm involving morphology [10].

Additionally, one shall not forget to mention the existence of algorithms that involve Hough transformation [2,39,83], which results in a set of line segments, that can be further processed [82]. In this chapter, I introduce an algorithm that uses Hough transformation for the localization task.

(a) Original image (b) Gaussian smoothing (c) Unsharp masking

Figure 2.1: Unsharp masking.

2.1 Pre-processing

Digital images acquired from a camera often need pre-processing due to device flaws or environmental difficulties. In images having low contrast, intensity levels should be normalized. In most of the proposed algorithms in this thesis, unsharp masking [57] is used for enhancement (Fig. 2.1), which is the weighted addition of the original image pixel intensities to the negated pixel values of the Gaussian-blurred version of the image. Compared to deconvolution, unsharp masking is convolution by a Dirac delta minus Gaussian smoothing kernel, and thus require less parameters and no model including aperture and focal length.

Image resolution for barcode localization usually does not have to be high, 1D barcodes having the narrowest line of two pixels is sufficient ($3 \times 3\,\text{px}$ median filters can be applied to eliminate salt-and-pepper noise). Higher resolution produces better results, but also increases computation time. The least time-consuming solution for downsampling high resolution images is the nearest neighbor interpolation, which is, besides being fast, is also a good choice because it preserves hard edges.

Some features (e.g. number of black and white pixels, zero-crossings) need to be extracted from binary images, so, as it logically follows, thresholding is also necessary in such cases. A simple threshold is sufficient on images with even lighting; otherwise adaptive threshold [81] is required.

Color information could also be taken into account for feature extraction, however, most visual codes maximize the contrast by using only black (or dark blue) and white colors. Furthermore, hardware used by industries are often set to operate and record only in specific color ranges. Taking the aforementioned facts into account, only intensity values are processed and color information is dropped in most cases.

In addition, it shall be pointed out that images affected by heavy noise also need

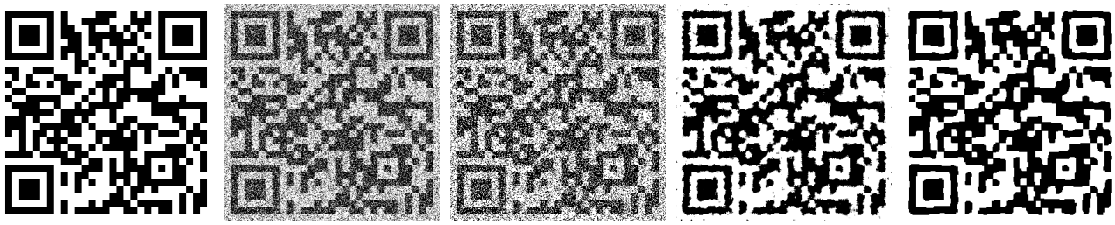


Figure 2.2: Preprocessing of a QR code. From left to right: Original image, image with 75 % uniform noise, after contrast enhancement, median filter, and morphological gradient.

multi-stage pre-processing with point operations, median filter or even morphological operations (Fig. 2.2).

2.2 Localization with scan-line features

The basic approach for barcode detection is scanning only one, or a couple of lines of the image (Fig. 2.3(a)). Scanned lines form 1D intensity profiles (Fig. 2.3(b) and (c)). Barcode detector algorithms [1, 37, 76] work on these profiles to find an ideal binary function that represents the original encoded data. The main idea is to find peak locations in blurry barcode models, then thresholding the intensity profile adaptively to produce binary values.

This concept imitates the process how a laser scanner sweeps through an image, scanning lines. The advantage of this technique is time efficiency and low hardware requirements. In most cases, only a subset of image pixels are read, thus a small buffer capable of storing a few image lines is sufficient. However, if we prefer to do more scans of the same data using differently oriented scan-lines, the whole image needs to be stored until the algorithm finishes. It is also important to mention that the approach often requires multiple scans of the same scene due to the relatively low accuracy.

Decoding part takes place on those processed profiles to recognize character sequence with various approaches. One example is a Bayesian decoding, with 4 directional scan-lines and entropy filtering [72]. According to Tekin et al., this type of localization is appropriate for mobile phone applications, as they proposed in later works [73].

Several experts assume in their work that barcodes are aligned along the axes [27, 71], or they are already localized [79]. Works of these authors propose accurate algorithms for decoding, that can be paired with proper localization approaches to

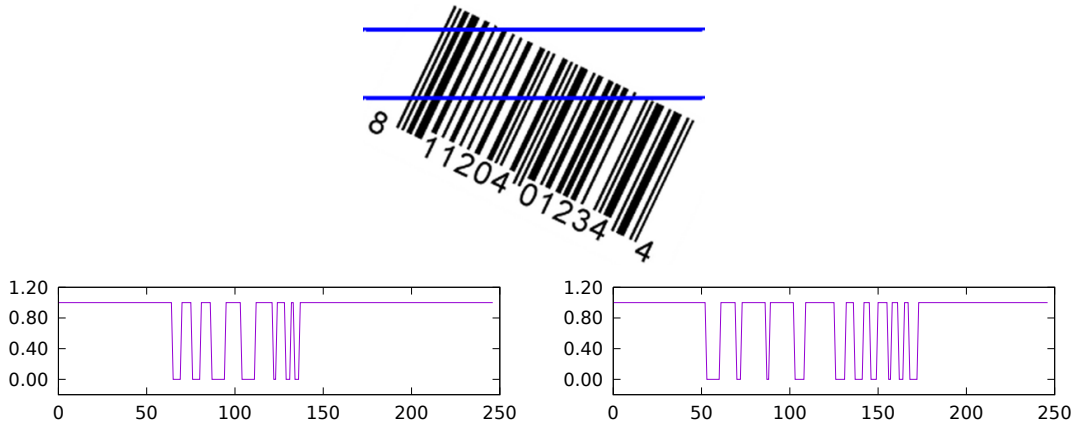


Figure 2.3: The idea of scan-line analysis. Scan-lines sweep through the code, finding areas with frequent intensity changes (ROI). ROI endpoints are barcode contour points. Each scan-line makes an 1D intensity profile.

solve the task of efficient barcode reading. For instance, Gallo et al. [27] also uses gradients for localization [63], which is derived from the scanned intensity profile.

Valley tracing (or bar tracing) [58, 76] is another low-cost method for finding barcodes in blurry, low resolution images, mostly on live smartphone camera frames. It consists of three steps. First, it finds starter points on the pictures, then follows the gradient ("valleys"), and finally, recognizes the endpoints of the bars. Moreover, there are also some scan-line based algorithms that involve the spectrum [19], or perform the analysis locally [35], using a tessellation of the image. My proposed modifications for scan-line analysis performed using image tessellation are presented in Chapter 3.

Before scan-line analysis is performed, a couple parameters have to be set, for example, the number of different directions, the density of the scan-lines and the method of post-processing. Some algorithms process scan-lines individually and look for valid barcode intensity sequences, while other methods only mark regions of interests (ROIs), and process the selected areas further (Fig. 2.4). This can occur in case of using more dense scan-lines, building of density images [84], line segment clustering [10], or more sophisticated algorithms, like the ones involving Hough-transformation [10]. The latter group is more efficient, since it does not solely rely on one single scan-line, and gives the opportunity to run more complex approaches on a significantly smaller data sets than the original image.

In this thesis, I propose using three or four directions for scan-lines, and instead of building density images that need many point-operations, I recommend grouping of lines by proximity and orientation as post-processing.

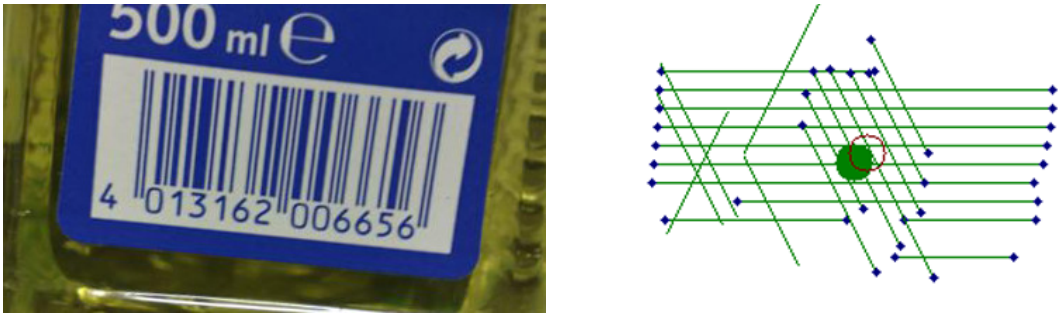


Figure 2.4: Scan-line analysis of real-life example. Original image (left) and the feature image (right), with line segments, endpoints, center for the horizontal line group (green, filled disc), and center for the 60° line segment group, dropped due to proximity to existing center point.

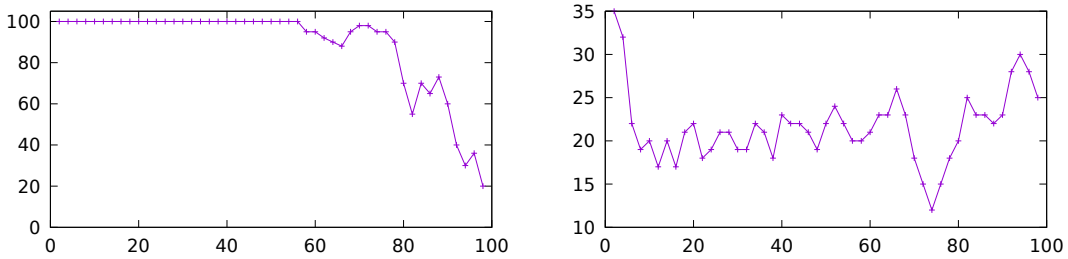


Figure 2.5: Detection accuracy (percent) and mean of detected center point distance from original location (pixels) regarding scan-line density.

The parameter of maximum step size (lowest scan-line density) that keeps accuracy high, was found out empirically. The smallest code for the test had a height of about 60 px, and test results show (Fig. 2.5) that distance does not affect detection accuracy until half of the code height. Around 30 px line step, results start to degrade. Visually it means that every code can be scanned as long as at least 2 scan-lines run across it. This rhymes with the sampling theorem. Scan-line analysis is also inaccurate (up to 20 pixels) for finding exact center points, which property comes from the density of the scan-lines.

A code candidate in this case, is a set of endpoints detected by scan-lines, and each point has a maximum distance from the center of mass and from the closest neighbor. When a scan-line finds ROI bigger than expected barcode height, the algorithm checks through the list of previously found barcodes and tries to add the ROI's endpoints if proximity conditions are met. If points cannot be added to any existing code, a new code object is created. After finishing the scan, code objects containing less points than a threshold are dropped. The threshold is defined as

Table 2.1: Accuracy of Scan-line analysis with and without unsharp masking (mean ± sd, expressed in percent).

Type	Size	Accuracy (mean ± sd)	
		without u. masking	with u. masking
Codabar	188 × 100	55.2 ± 42.9	97.2 ± 16.4
Code 11	176 × 64	100.0 ± 0.0	100.0 ± 0.0
Code 128	180 × 150	51.1 ± 40.2	88.5 ± 31.9
Code 39	332 × 100	69.6 ± 34.4	98.4 ± 12.7
EAN-13	190 × 138	86.2 ± 22.2	99.5 ± 7.1
Plessey	402 × 80	58.6 ± 38.0	94.6 ± 22.7
UPC	187 × 96	94.7 ± 10.5	100.0 ± 0.0
UPC-A	190 × 120	95.2 ± 8.5	100.0 ± 0.0
All together		84.1 ± 23.5	97.3 ± 4.8

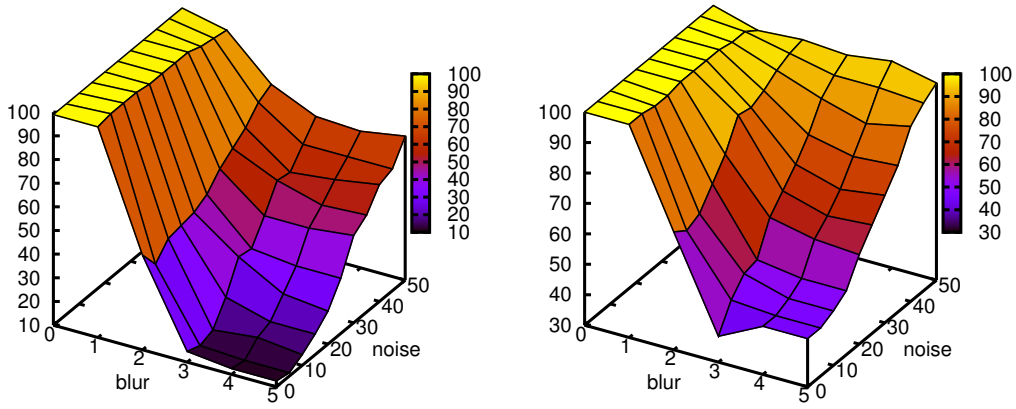
$T = \max(4, \text{barcode height/line step in pixels})$.

For example a 120×60 px code with a 20 px line step needs to have at least 6 endpoints found by line scan. Scan-line analysis with multiple oriented scan-lines can find "well-oriented" barcodes multiple times, so I introduced a repulsive factor for the momentums of codes. Every code can invalidate other codes that have fewer endpoints found and have a momentum closer than half of the barcode height.

Scan-line analysis is a fast method, however, it is very sensitive regarding orientation. Working with only 2 perpendicular scan-lines produce poor results, especially on codes having more elongated shape (Plessey codes). 3-scan-line analysis often treats "badly-oriented" Plessey codes as two different smaller codes. Working with 3 scan-lines $(0°, 60°, 120°)$ reduces the miss rate to an acceptable level. Altough, when scanning for barcodes with 1.5 to 2 width-to-height ratio, 3 scan-lines seem sufficient, for codes with bigger ratio, 4 scan-lines are recommended $(0°, 45°, 90°, 135°)$. In extreme cases, like in the case of noisy images or reflection artifacts, it makes sense to increase the tolerance for endpoint proximity or the threshold of minimum number of points in a code.

Test results of the scan-line approach (Table 2.1) show that detection accuracy varies for different code types. These accuracy bounds depend on line density and width-to-height ratio of the codes, besides the amount of noise and blur (Fig. 2.6). The application of unsharp masking seems to improve detection accuracy considerably for all types of barcodes.

Scan-line analysis can also be used for pattern matching the locator regions of 2D codes. Due to the omnidirectional nature of most locator patterns, only scan-lines



(a) (b)

Figure 2.6: Detection accuracy of Scan-line analysis with respect to blur and noise level for 3 (a) and 4 (b) scan-lines. Applying 4 scan-lines show significant improvement on noise tolerance.

in one direction are required. Knowing the element size of codes, we can look for black (B) and white (W) runlengths in each line, for example QR and Micro QR has BWBBBWB locator pattern, Shotcodes have BWBWB, Aztec codes have WB-WBWBWBW (or BWBWBWBWBWBWB, depending on embedded data size), Maxicodes have BWBWBWBWBWB (Fig. 2.7). For circular bulls-eyes (Maxicode, Shotcode), scan-lines of any orientation can detect the exact size of pattern line widths, while with rectangular locator regions, 1 to $\sqrt{2}$ times pattern line widths should be expected. Codablock codes, PDF417 codes and Data matrices do not contain omnidirectional locator pattern, so they cannot be detected with this approach. However, Codablock and PDF417 codes have a well-defined locator pattern at the beginning and at the end of codes, which scan-lines can detect from a certain direction. In this case, direction and density of the scan-lines are also important parameters. For other code types, I recommend a dense line-scan of the image for finding the exact position of potential locator patterns. Scan-lines of one direction are sufficient for this task. Data matrices have similar BWBWBW locator patterns inside the embedded data.

In general, although scan-line algorithms are fast and easy to implement, they show low accuracy, which renders them unreliable in industrial environment.

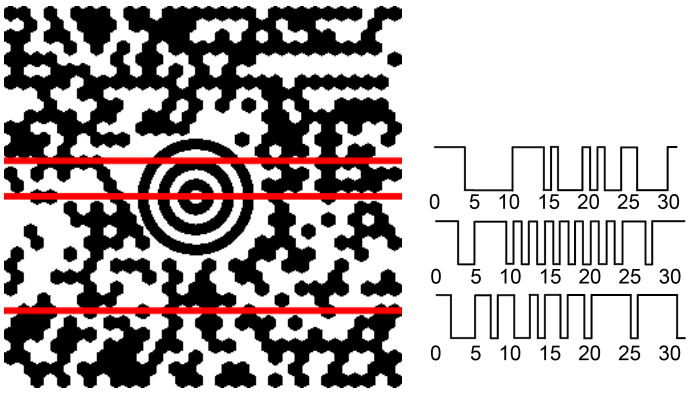


Figure 2.7: Scan-line analysis on a synthetic Maxicode and some of its intensity profiles.

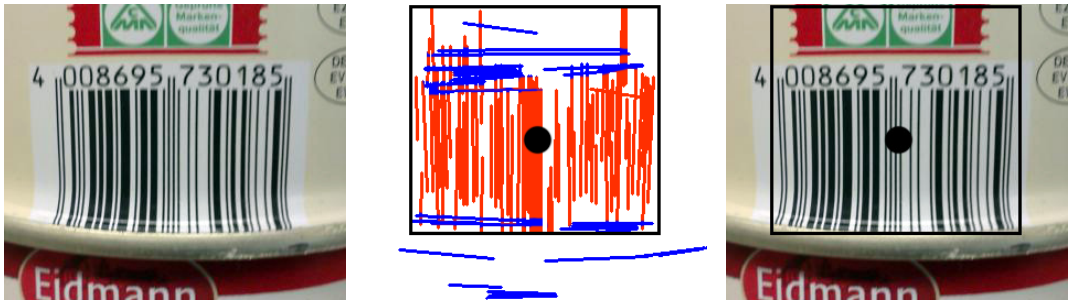
2.3 Localization using Hough transformation

Several algorithms involve *Hough transformation* [55,60,67,82,83], a tool for shape and pattern recognition [2]. Hough transformation is capable of localizing barcodes, as they consist of roughly equally long, parallel line segments in a small area. The transformation needs an edge map as input, usually produced by Canny edge detector [18], or convolution, using Sobel kernels.

The two most common variants are *standard* and *probabilistic Hough transformation* [39]. Both variants transform edge points into Hough space first, and compute line locations, lengths, orientations, and proximity to each other [5,82]. Although both edge detection and transformation of the edge points to Hough space, which are regarded as slower steps compared to basic scan-line methods, the approach does not require assumptions on the orientation of the code object [48], and it is more tolerant towards noise and smoothing. Furthermore, Hough transformation can be used as an intermediate step of another barcode localization approach, but I also evaluated it individually as a feature producer.

For pre-processing, I recommend a blur filter on high quality images, since smooth images are desired by the Canny edge detector. The aforementioned attribute makes Hough transformation especially convenient for images already having some level of smoothing.

After the Hough transformation, a list of line segments with their center points, length, and orientation is obtained. Those lines are then grouped to decide whether they constitute a barcode or not. This step, however, has multiple parameters. For instance, minimum number of lines, the proximity needed for the lines to be in the same group, and the tolerance for length and orientation from the means within the



(a) original image (b) feature image (c) overlay

Figure 2.8: Canny edge detector with Probabilistic Hough transformation. In (b), detected lines that are part of a barcode-like cluster are shown in red while the other detected lines are shown in blue. (c) shows the detected bounding box and centroid of the code region overlaid onto the original image.

group. These parameters have to be adjusted in each application experimentally. As barcodes tested for this thesis consist of at least 25 parallel lines, the minimum number of lines defined was 20, to give reasonable tolerance.

In the final step, group centers are returned, and the image can be cropped for decoding with known barcode decoding implementations (Fig. 2.8).

Using the Hough transformation only for localization is faster than using it as part of more sophisticated algorithms. It shall be emphasized that in order to achieve higher efficiency of the algorithm, it is necessary to set its parameters accordingly. Moreover, Hough transformation even detects the dominant orientation of line segments within the ROI, thus giving a compact solution for the localization task.

2.4 Localization using mathematical morphology

A different approach towards barcode localization is to filter shapes and observe texture-like properties and detect properties that refer to barcode-like appearance. Algorithms with mathematical morphology [5, 10, 36, 38, 47] use the combination of basic morphological operations like erosion and dilation. White blobs on these intermediate feature images show the possible barcode locations. Further processing, like segmentation and filtering of small components are taking place after that step.

This approach can be applied on both 1D and 2D barcodes as well. Algorithms of this chapter are considered as the slowest method of barcode localization, since convolution with large kernel size is a bottleneck at processing high-resolution images.

However, convolution can be parallelized, and high accuracy of these algorithms make them qualified for industrial use.

2.4.1 Bottom-hat filtering

Juett and Qui [36] developed an algorithm based on bottom-hat, or black-hat filtering. The pre-processing step of this algorithm requires correcting intensities of the non-ideal image with contrast stretching. Moreover, color to grayscale conversion is also performed if necessary.

The step that follows is bottom-hat filtering, that is, when the closing of the stretched image is subtracted from the original stretched image ($T_b(f) = f \bullet b - f$). The structuring element size depends on the widest bar in the barcode to be detected. In their paper Juett and Qui [36] specify 25×25 block structuring element (SE) for images of size 720×480 pixels.

Additionally, when talking about less complex images, the false areas are fewer after bottom-hat filtering than after gradient calculation. The rest of this algorithm works with binary images, hence the next step is the binary conversion.

After the filtering, the contour is defined. The binary image is eroded using a 5×5 SE and subtracted from the original binary image. This is followed by the determination of the orientation, which is performed by directional image openings using a relatively large linear SE. These openings are performed at 16 different orientations, with a step of 11.25°. Due to the fact that the barcodes could be in any orientation between 0° and 180°, selecting one suitable orientation seems to be a difficult process. For this very reason, several orientation is probed which significantly increases the execution time of the algorithm.

The directional opening images are summarized and a low resolution density image is calculated, which is followed by converting the image back into binary. In this case and each region represents a potentially barcode region.

In the last step, objects with a smaller area than a threshold are eliminated. Using the centers of the remaining objects, lines that are next to each other are found and the potential corner points of the object are computed.

2.4.2 Hit-or-miss transformation

In Tuinstra's algorithm [76], the authors rely on that in the barcode region, the intensity difference between the stripes is high, and as a result of this, a gradient calculation highlights the bars. To estimate the gradient in the x and y directions,

Sobel kernels are used. This process is followed by thresholding the gradient image and selecting pixels having a high gradient value.

The rest of the procedure works on the binary image. The first step is the hit-or-miss transformation with a line-shaped SE, as $A \circledast SE = (A \ominus C) \cap (A^c \ominus D)$, where (C,D) is the composite structuring element. Unfortunately, the SE is not specified in the article [76]. In my implementation, the SE was chosen considering the length of the longest bar in the images. A 10×10 px SE was used, which is sufficiently large for the objective and smaller than the expected bar length in the codes. In the course of this procedure all objects are removed that are not likely to contain a barcode.

The second step of the algorithm is morphological dilation. This is performed in order to merge nearby but not necessarily connected objects to be able to compose a region. The SE is square shaped, but its exact size was not specified in the article [76]. I used a 10×10 px block SE matching the size used for the previous step. In practice, the image size determines this parameter. In addition, as it may be predicted, dilation fuses some regions that do not contain a barcode, however, such results are eliminated in the subsequent phase.

The third step consists of performing morphological erosion to discard thin objects from the image. In this step, the SE size is greater than the one applied in the dilation process. A 20×20 px block SE was suitable for setup in this thesis. This step removes undesired segments which were fused by the dilation.

The last step is a solidity test which compares the number of pixels turned on in a region to the convex hull of the region. Therefore, this step removes false positive objects while only barcode regions remain.

2.4.3 The MIN-MAX method

The name of the *MIN-MAX* method proposed in this thesis [45] originates from the intermediate steps of the algorithm itself, more precisely, from the grayscale erosion and dilation that produce "minimum" and "maximum" images via local infinums ($(f \ominus b)(x) = \inf_{y \in B}[f(x+y) - b(y)]$) and suprenums ($(f \oplus b)(x) = \sup_{y \in E}[f(y) + b(x - y)]$). As this method manages well noisy, blurry or distorted images [10], apart from an optional normalization, pre-processing operations are not required. Moreover, *MIN-MAX* does not require directional openings either, making this approach faster compared to the aforementioned one.

Taking the assumptions about the maximum bar width of a barcode into consideration, for the tests carried out for this thesis, the morphological gradient operator

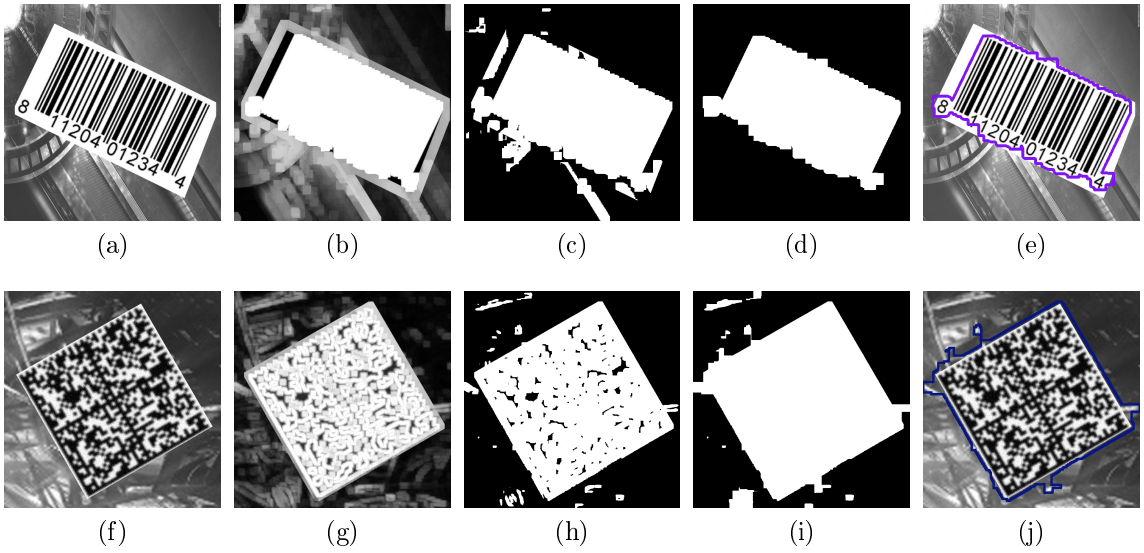


Figure 2.9: Stages of MIN–MAX method. (a): original image; (b): morph. gradient; (c): binary threshold; (d): opening; (e): contour overlaid to original image; First row: EAN-13 barcode (a-e); Second row: Data matrix (f-j).

($G(f) = f \oplus b - f \ominus b$) was applied on the image with a box kernel with the width of the expected widest bar plus one unit (Fig. 2.9(b)). Next task is to remove "weak" shapes from the feature image with a binary threshold (Fig. 2.9(c)). A good rule of thumb regarding the threshold can be at 75 % of the full intensity scale since barcodes produce areas close to the maximum intensity.

After the application of morphological gradient operator and the thresholding of the result, morphological opening operation if performed on it (Fig. 2.9(d)), with the previously defined kernel for closing the small gaps caused by scratches, reflections or other flaws of the original image.

At this stage the feature image already indicates the barcode-like areas with white, thus, the last operation that shall be carried out is measuring the exact area of these areas (Fig. 2.9(e)), and their centers. Experiment of this thesis conducted that setting the minimum area to be classified as barcode to $w \times h \times 0.75$ or lower is satisfactory (where w and h denote for barcode width and height respectively), but this parameter has to be adjusted in accordance with the end-user application.

2.5 Evaluation and results

There are several barcode detection software and frameworks on the market, like the DTK Barcode Reader SDK[1], BC Tester[2] and Barcode Recognition SDK of DataSymbol[3]. Although they do not indicate the applied algorithm behind their detection mechanism, a brief qualitative comparison has been made and the results are shown in Fig. 2.10 together with the results of the experiments conducted for this thesis (see Fig. 2.11).

Results show that Datasymbol has the most robust localization approach, while BCTester had very poor accuracy even with clean and sharp images. The Tuinstra et al. and Juett et al. algorithms worked reasonably well, both has performed at least on the level of commercial softwares. Hough transformation is more tolerant to blur than noise, while MIN–MAX seemed to be the most robust approach.

For comparing the effectiveness of the presented methods, the most common measures such as, precision, recall, accuracy and F-measure (Eq. (2.1)) are used. The values are based on the Jaccard index (Eq. (2.2))

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.1}$$

$$J(G, D) = \frac{\sum_{x,y}(G(x, y) \wedge (D(x, y))}{\sum_{x,y}(G(x, y) \vee (D(x, y))} \tag{2.2}$$

where G and D give binary 0–1 values based on the pixel intensity of the ground truth and the detector output images (both are binary).

The average performance indicators of the detectors are shown in Table 2.2. The Probabilistic Hough method is a robust choice to localize barcodes, because it can be parameterized to minimum line length and maximum gap between line segments. It also handles noise well to a certain level, but it is quite sensitive to distortions. The proposed MIN–MAX variant is very tolerant to noise and smoothing, due to its morphological approach. However, the convolutions used in the steps of the algorithm make it relatively slow.

[1] http://www.dtksoft.com/
[2] http://www.bctester.de/
[3] http://www.datasymbol.com/

Figure 2.10: A qualitative test of barcode detectors from the state of the art and the market. From top to bottom: original image, Tuinstra et al. [76], Juett et al. [36], DataSymbol software, DTK barcode reader, BC tester. Each column represents a different test image.

Figure 2.11: A qualitative test of barcode detectors based on our features. Contours of the possible barcodes are shown in red. From top to bottom: original image, Hough-transformation, MIN–MAX. Each column represents a different test image.

Table 2.2: Average detection performance of the presented methods.

Algorithm	Precision	Recall	Accuracy	F-measure	Runtime
Tuinstra et al. [76]	57.08 %	85.29 %	84.19 %	48.39 %	160 ms
Juett et al. [36]	34.26 %	94.08 %	72.76 %	36.13 %	230 ms
Hough trans.	64.83 %	85.07 %	84.22 %	58.76 %	230 ms
MIN–MAX	43.36%	85.38 %	77.47 %	36.82 %	360 ms

2.6 Summary of the chapter

In this chapter, algorithms that use global information for the visual code localization task were presented. The oldest, simplest member of this group is the algorithm family that imitates scan-lines. Such algorithms seem to lack adequate accuracy and are sensitive to noise and smoothing. Furthermore, the experiment conducted with the application of the Hough transformation, in order to extract line segments and mark code candidates was also presented. The most reliable algorithm family is the one involving mathematical morphology. Algorithm improvements and proposals of this chapter were published in [10].

Chapter 3

Algorithms based on image tessellation

End-user applications mostly have limited resources, like memory and processing power. Not all hardware setups let to store the whole image in memory, some even drop information as new sensory data is acquired, thus providing only partial data by time. In this chapter, I present methods that work with image tesselations and make decisions based on local cell information [6, 10, 35]. For pre-processing, the same remarks apply that are discussed in section 2.1. After the low-level processing of cells, a feature image is assembled and processed on a higher level.

Speed is a primary or secondary objective in most cases. Algorithms of this chapter use simple features and thanks to the local measurements, they can be parallelized easily.

Image tessellation (partitioning the image to uniform cells) is a wide-spread idea of pattern recognition, which can be used as the base of barcode localization. This approach is applicable because most barcodes, just like areas covered with textures, can be easily identified by observing only small parts of them. These barcode parts together form the desired barcode region with known height and width.

It is important to note that each cell is assigned a value that indicates the grade of the barcode presence, however, the choice of the feature varies. As the result of the evaluation, a matrix is formed from these values, each element representing a cell value. This matrix can be also considered as a low resolution feature image.

Texture parts have similar local statistics in their neighborhood, hence, searching this matrix for compact areas of similar values defines regions of interest, representing barcodes with high certainty. Furthermore, it is important to mention the compactness of code parts, which attribute helps to filter out small cell groups or

groups having low compactness.

The feature matrix is thresholded in most cases, and processed via connected component labeling [22], which can be performed rapidly, using the algorithm of Suzuki et al. [68].

Algorithms proposed in this chapter examine the image in small, disjoint or overlapping cells, and make local measurements, some even consider neighboring cell information (Section 3.1.1). Local variants for the basic scan-line analysis are also introduced as well (Section 3.1.2 and 3.1.1). New algorithms using distance transformation and cell histograms are also proposed. These algorithms can be used for fast localization, however, parameters and features have to be chosen properly so as to maintain accuracy at a reasonable level.

3.1 Extending the scan-line approach

In classical scan-line approach, scan-lines sweep through the whole image. In this section, the scan-line idea is adapted to small image cells, where pixels are read along various patterns.

3.1.1 Local scan-line with circular pattern

Replacing the classical line-scanning method is one of the improvements I propose in this section, published in [13]. The modification of the scan-line approach with circular pattern involves the subsequent procedure. As the beginning, it is required to convert the image to binary by thresholding. Binary images are divided into square tiles with overlapping by half the tile size. Each tile is processed individually at first, and a measure is assigned by evaluating pixels in a circular pattern, with the tile size as diameter. A one-dimensional profile is obtained, which has zero-crossings of various densities.

After this step, the circle forming the intensity profile is divided into four equal-sized quadrants by density of zero-crossings. Image parts representing a barcode have equally low or high number of crossings at opposite quadrants, and significant difference in the number of crossings at neighboring quadrants. For the sake of simplicity, quadrants having many and few zero crossings are labeled as “Wild” and “Calm”, respectively. Those quadrants can be defined and separated, as shown in Figure 3.1.

Wild quadrant centers of a tile are placed by maximum of the following formula:

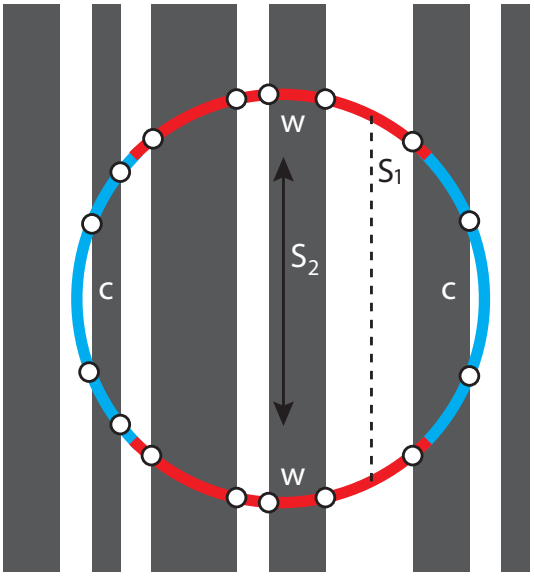


Figure 3.1: Zones and symmetries of the circular intensity profile. Wild (w) and calm (c) zones, symmetry at pixel level (S_1) and between quadrants S_2.

$$D(T) = \max\left(\frac{1}{2q}\left((Z_{w1} + Z_{w2}) - (Z_{c1} + Z_{c2})\right)\right) \tag{3.1}$$

where Z holds for zero-crossing, w and c for wild and calm zones respectively, and q for quadrant size in pixels. D shows the direction of the first wild quadrant in the best fitting placement of quadrants along the circle. According to the symmetry of the circular pattern, $D - 180°$ is applied, where D is greater than 180 degrees.

Connecting the wild zone centers, defines a dominant direction of the possible barcode pattern in each tile, and neighboring tiles will have the same dominant direction when containing barcode parts. Additionally, taking the strength of the dominant direction into consideration is of essential importance. The maximum at Eq. 3.1 is small at tiles having very small or high level of intensity entropy, and there can be multiple possible dominant directions as well.

Due to image flaws, noise, reflections and distortions, a level of tolerance should be introduced at neighboring tiles. Within that range, dominant directions should be considered as equal. Furthermore, having all neighboring tiles with similar dominant directions and different from the examined tile should suppress the assigned direction and make the alignment reconsidered.

Additionally to the aforementioned features, symmetries are also well-traceable attributes within a cell, thus can be used for searching barcode parts. The level of symmetry at pixel level and between quadrants are computed with Eq. 3.2 and

Eq. 3.3, respectively. These symmetries can only be detected at wild quadrants due to the axial symmetry of barcode patterns.

$$S_1(T) = \frac{1}{q}\sum_{i=0}^{q} 1 - |W_1(i) - W_2(q-i)| \tag{3.2}$$

$$S_2(T) = 1 - \frac{|Z_{w1} - Z_{w2}|}{q} \tag{3.3}$$

The smallest difference in dominant direction at T and its 4-adjacent neighborhood T^* as expressed by Eq. 3.4 are searched for, and the final measure assigned to each tile is by Eq. 3.5. As the weight of D_m, δ is applied, since neighborhood similarity is more important than symmetry in tiles ($\delta = 2$ is the proposed value).

$$D_m(T) = 1 - \frac{1}{90}\min\{U \in T^* \mid |D(T) - D(U)|\} \tag{3.4}$$

$$V(T) = \frac{1}{2+\delta}\left(\delta D_m(T) + S_1(T) + S_2(T)\right) \times D(T) \tag{3.5}$$

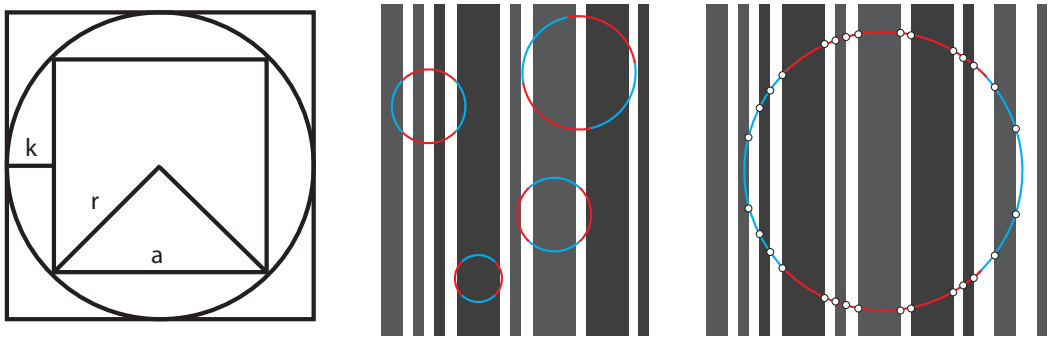
The feature matrix is formed by $V(T)$ values. The thresholded matrix is processed as described above. Convexity of the returned blobs could also be examined, however, it would decrease the localization performance. Since low rate of false positives are not crucial, a better solution is to let the decoder decide whether concave regions contain valid barcode.

As it is fully direction-invariant, scanning the image parts in circular pattern instead of a set of lines with fixed number and orientation is concluded to be more efficient. Moreover, it uses symmetry to reinforce decision, and processes just a fraction of pixels within a cell.

Downsampling of high resolution images having low noise level, good contrast and crisp edges is allowed and recommended, but all setups should be examined for minimum resolution of the input in order to achieve accurate localization. In idealistic images, like the artificially rendered ones, 1 unit[1] as 1 px is satisfactory, however, at least 2 px as unit size is recommended for real-life applications. Strict industrial setups can be considered as idealistic, as they have even illumination and quality camera. The proposed downsampling to obtain desired unit size is the nearest neighbour interpolation, since it preserves hard edges of barcodes.

The optimal tile size here is about 25 units. UPC-A codes have a total width of

[1] In patents, barcode dimensions are often expressed in units, a resolution-independent representation relative to the smallest bar width.



(a) (b) (c)

Figure 3.2: (a): Geometry of a tile and zone widths; (b): badly chosen tile sizes; (c): well-chosen tile size, 26 units, worst case.

95 units and a height of 78 units. The upper bound to tile size is 36 units, since larger tiles would not fit along the smaller dimension of the code, and thus reducing accuracy dramatically, while lowering the minimum number of tiles required to form a barcode to compensate the effect, would raise false positives.

The worst case is the one when digits 6 and 3 are besides each other, which causes 1-4-4-1 width pattern. As it is illustrated in Fig. 3.2a, wild zone width is $\sqrt{2}\times r$, and calm zone width is $1/2\times r(2-\sqrt{2})$. Moreover, at Fig. 3.2b, badly chosen tile sizes are present. The zones are eventually defined well (top left circle), but the same sized circle fails at thicker bars (bottom right circle). In order to ensure that the worst aligned tile is able to indicate sufficient zero-crossings to align wild and calm zones correctly, tile size has to be at least 13 units.

In accordance with these to these bounds and by letting the maximum variance in expected barcode size while keeping high accuracy, 24–26 units is the recommended tile size (Fig. 3.2c). Computing with an optimal case where calm zones contain no zero-crossings at intensity values, leads to the same conclusion, since width of k using 26 units as tile size is $(26-13\sqrt{2})/2$, which is about 4 units, the thickest bar width of UPC-A patent. However, zero-crossings in the calm zone are not problematic over the lower bound of tile size.

A more accurate replacement of finding the dominant pattern direction is the fast Hough transformation. It provides the most reliable results; however, it slows down the localization algorithm to a level comparable to morphological operations.

Only minimal trigonometric calculations are necessary at the initialization step, since discrete relative coordinates to the points of the sampling circle have to be calculated only once, and can be applied by offsetting with center positions. Fur-

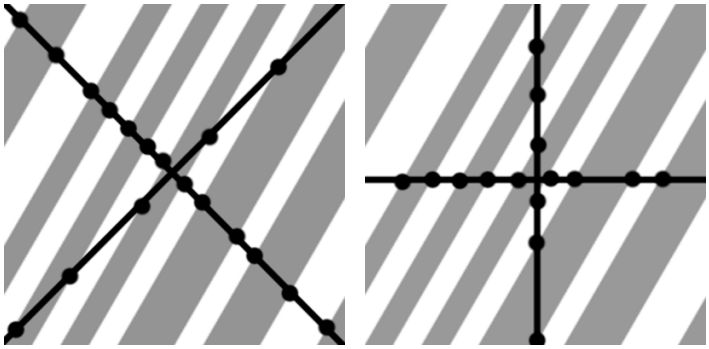


Figure 3.3: Two pairs of scan-lines sweep through the image. One of them has significantly higher number of zero-crossings in case of positive response. In this example, the barcode is fully recognized by the first pair of scan-lines.

thermore, if a map is stored with the corresponding angle to each relative coordinate, calculation of the dominant direction does not require arcus tangent.

3.1.2 Runlength measuring

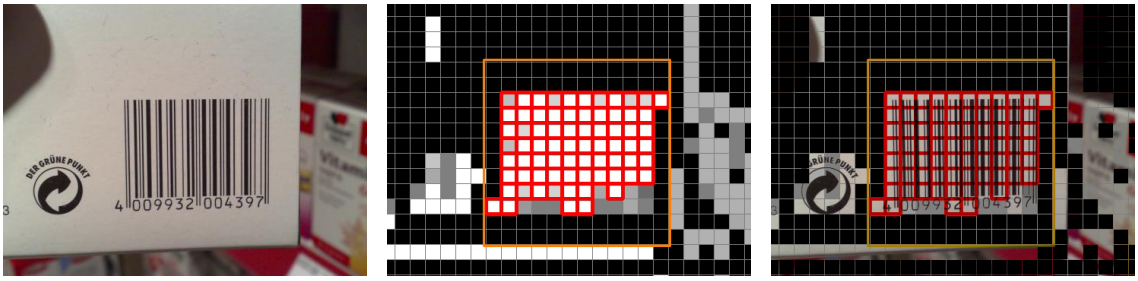
Runlength Measuring is a tile-based local scan-line approach. It requires using a well-chosen tile size, each tile is examined using two pairs of perpendicular scan-lines, one pair at 0° and 90° and the other at 45° and 135° (Fig. 3.3). A measure is derived from the difference of the count of the intensity changes along the scan-lines (Eq. (3.6)). For instance, a horizontally aligned barcode has a lot of intensity changes when scanned with a horizontal scan-line, but has few or none with a vertical (Fig. 3.3).

With the 2 pairs of scan-lines, barcode pieces of any orientation can be safely recognized. The final measure assigned to a tile is the maximum of the two differences. This measure gives 1 if parallel lines are present on an image tile, and 0 if a tile contains a homogeneous area, or noise. The real-life example and feature image can be observed in Fig. 3.4.

$$C_i = \frac{|IC_{i1} - IC_{i2}|}{\max(IC_{i1}, IC_{i2})} \tag{3.6}$$

where IC_{ij} is the intensity change count of a scan-line j in scan-line pair i.

Optimal tile size was determined experimentally. Since the smallest barcode has a 60 px height, 30×30 px or greater tile sizes have poor recognition capability. Moreover, since the characters appearing below the code and code parts nearby also have a barcode-like property (plain text is not affected), a very small tile sizes also



(a) (b) (c)

Figure 3.4: Runlength measuring on a real-life example. Original image (a), feature image with bounding box (b), overlay (c).

lead to greater error for computing the center of the codes. The best tiling size is about 1/3 of the barcode's height.

It shall be also mentioned that it is, indeed, possible to run this method with disjoint or overlapping tiles, but overlapping does not improve the method's accuracy; it only increases computation time. Furthermore, offsetting the tiles has no significant effect, since it just produces some blocks to be more and others to be less "barcode-like" at the barcode's opposite sides.

Runlength measuring is better than the basic scan-line analysis because it relies on a large amount of cell values instead of only one global feature value. Despite the fact that, this method might result in low certainty values in the feature matrix in case of blurry and noisy cells, it still gives ROIs if it finds sufficient number of positively labeled cells around.

Regarding 2D codes, although accuracy can be improved using more scan-lines, runlength measuring is not considered to be reliable. As for an example, six local scan-lines sweep through the tile, three horizontally and three vertically (Fig. 3.5). The intensity changes are counted and summed in both directions. If these two sums are close to each other, then the degree of intensity change is roughly the same both horizontally and vertically.

It shall be mentioned that many barcode types have this feature, for instance, the Aztec code, Shotcode, Data matrix, QR and Micro QR code. Even PDF417 and Maxicode can be detected with limited accuracy by the application of this feature.

3.2 Local intensity clustering

Local pixel clustering [10] is the most simplistic algorithm based on image tiling. It divides cells into black and white segments. An image region that contains a

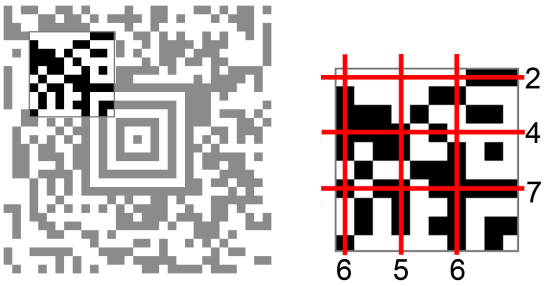


Figure 3.5: The idea of intensity change counting represented on a QR code.

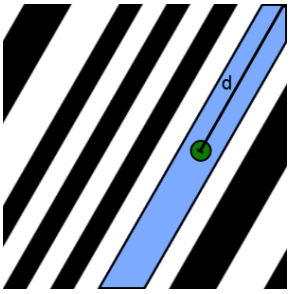


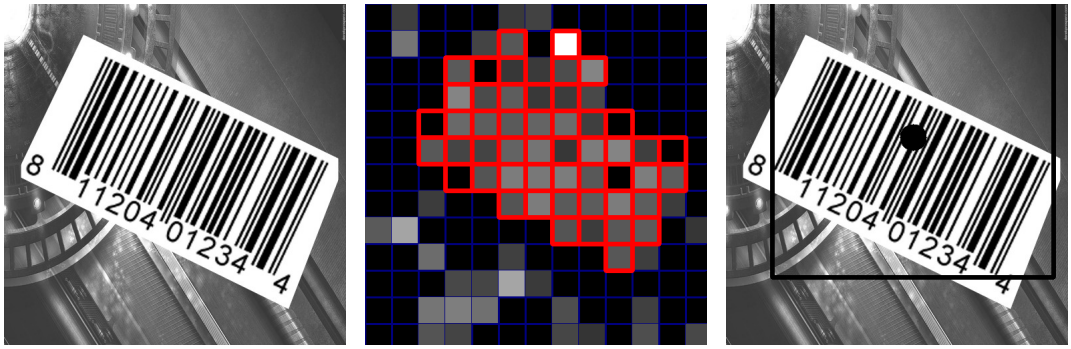
Figure 3.6: The idea of Local clustering. Here d is the maximum distance from the cluster center, i.e. the half of the cluster diameter.

barcode part has many similar stretched clusters (Fig. 3.6). The minimum count of expected segments can be derived from the widest bar of the barcode and the tile size. Similarly to most approaches of this family, parameters require assumptions on the expected barcodes.

Degree of stretch can be defined as twice the distance of the furthest point of the segment from its center. With exactly horizontal or vertical lines, the largest value of this measure is the tile size, however, in oblique situations, it is expected to be longer than that. Furthermore, stretched separate segments need to be aligned approximately identically. Otherwise, one segment would merge with another, decreasing the number of separate components in a tile and thus falling below our threshold.

When it comes to pre-processing, median filter is used first. On real-life images having low contrast at barcode areas, adaptive thresholding is necessary beforehand. Minimum segment size can be easily computed from the maximum bar width multiplied by the tile size.

After evaluating all tiles locally, the feature matrix is searched for compact areas (Fig. 3.7). Threshold is applied for values to classify whether or not an area contains a barcode part. Defining the threshold above $T = 0.5$ decreases detection accuracy,



(a) (b) (c)

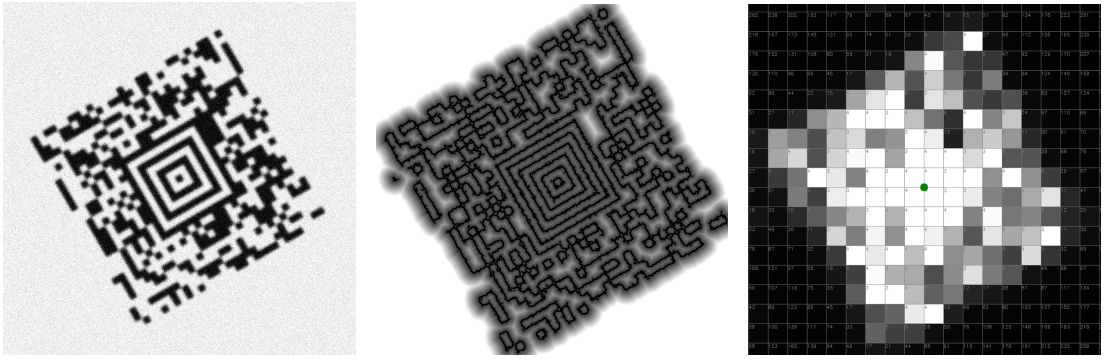
Figure 3.7: Stages of Local clustering: (a): original image; (b): feature matrix visualized as gray values. Red squares are above threshold; (c): code center and bounding box.

while setting below it increases false positive rate. After the connected component labeling, small components are dropped, and the centers of the remaining clusters are returned. Clusters are considered small when they contain less than N tiles (Eq. (3.7)), where s is the tile size (which is 1/3 of the code height in our examples).

$$N = \max\left(2, \frac{|h-s| \times |w-s|}{s^2}\right) \tag{3.7}$$

Bounding boxes in the output samples used in this thesis are not enclosing the whole barcode in every case. This is due to the fact that only the lower and upper bounds for the clusters in the feature matrix were calculated, and barcode corner pieces are too weak for the feature. Bounding boxes can simply be improved by finding aligned rectangles instead. The best tiling size is about 1/3 of the barcode height, like at runlength measuring (Section 3.1.2).

Running the method on the same scene with different offsets yield different detection accuracy [10], which shows that this approach is sensitive to the choice of tiling. Evaluation with overlapping tiles can be introduced as in two phases. In the first phase, *Local clustering* is performed with zero-offset tiling, and in the second phase the same procedure is done by the application of an offset of half the tile size in both directions. The code centers detected by the aforementioned phases are pooled together with an extra filtering wherein those code centers that are detected in both phases and are close to each other are merged into one (the larger cluster is kept), because they are likely to correspond to the same code.



(a) original image (b) distance map (c) feature image

Figure 3.8: Uniform partitioning of a synthetic Aztec code (a). On the feature image (c), each tile has a value of mean distances of the distance map (b). The feature image has darker "holes" where blocks with similar intensity are placed next to each other.

3.3 Distance transformation

This thesis also proposes an algorithm for localization based solely on distance transformation [25]. Distance transformation can be used individually with limited performance, or as an intermediate filtering step for creating more sophisticated algorithms.

The pre-processing step here is normalization of the intensity levels. After that, tessellation happens. Each cell is assigned a value based on distance transformation of the edge map. Distance transformation is an operation that works with an initial set of points, like, corners or edges. Value 0 is assigned to these points, and any other point gets the distance value to the closest point of the initial set. For this procedure, Canny edge detector is applied on this point set, providing input to the distance transformation. For each cell of the distance map, means and standard deviations are calculated.

As for 1D codes, distance values are spread between half of the minimum and half of the maximum line width, while, for 2D codes, these values usually stay below half of their block size, but higher values are also possible, since the blocks of the same color can be next to each other in multiple directions. Presence of such code parts significantly raise the mean of distance values within the image tile. However, these "holes" in the feature matrix are rare enough, consequently, they do not have an effect of splitting blobs of barcode-like areas (Fig. 3.8).

The evaluation of the feature matrix is same as in Local Clustering. Experiments

(a) original image (b) colored feature image

Figure 3.9: Sample image and feature image. The feature image is based on distance transformation of the edge map. Small clusters (shown in red), and shapes having low compactness (shown in blue) are dropped. Important regions are colored green.

conducted indicate that when it comes to 1D codes, mean of distance values spread around half of the average line width, while, for 2D codes, the values spread around the square root of their element size. Additionally, a threshold was applied for the values so as to classify whether or not an area contains a barcode segment. More importantly, letting a 25 % tolerance for these idealistic values, detection accuracy becomes satisfactory.

For end-user applications, noise, scratches and reflections should be taken into account. Regarding images containing heavy noise, distance means drop. Barcodes suffering from scratches, dust, handwriting or reflections, change the distance means significantly in accordance with the dark or bright intensity values of the flaw. Therefore, tolerance should be adjusted to these distracting properties and exact values can also be measured via trial and error. Tolerance value is a compromise between accuracy and the rate of false positive detection, thus, it should be set with respect to the final application.

The remarks of the above sections on cell size are also valid in this particular case. It is possible to run the partitioning with disjoint or overlapping cells, but one has to have in mind that overlapping does not improve the method's accuracy, it only increases computation time. Offsetting the cells has no significant effect either.

Being used individually, distance transformation gives a noticeable amount of false positives, since text can also satisfy well the condition of having similar distance means than a barcode. However, it can be used as a weak classifier of image areas, and its output is a good starting point for more accurate methods.

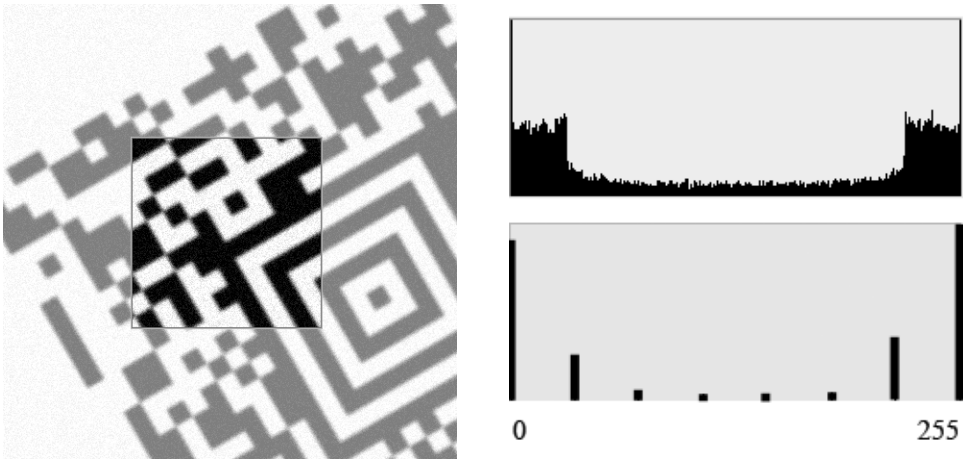


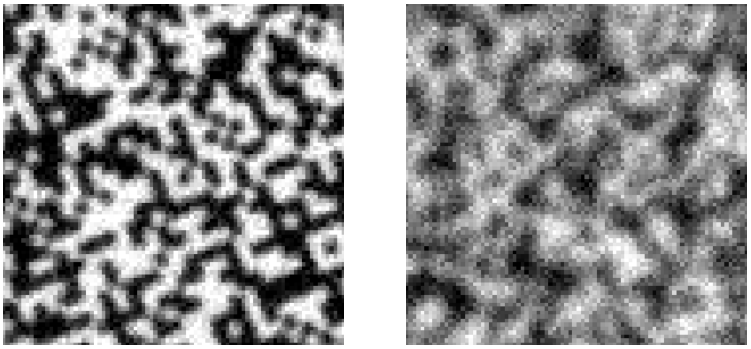
Figure 3.10: Aztec code part with 25 % uniform noise and Gaussian blur ($\sigma = 2$) (left), its 256-bin histogram (top-right), and 8-bin histogram (bottom-right).

3.4 Cell histograms

In case of an ideal cell containing a part of a visual code, only black and white intensities are present cell-wise in roughly 1:1 proportion. Due to the variability of the code object and imperfections of image acquisition devices, the intensity histogram deviates from this ideal case (Fig. 3.10). To take the effects of smoothing and noise into consideration, a formula can be introduced to model theoretical cell histograms.

Smoothing of images that contain QR code, introduces intensities closer to the mean, and decreases the value of extreme histogram bins, like the ones belonging to black and white. In general, the peaks are lowered, some bins that are close to the peaks receive higher values, while others keep their values. As a very simplistic approach, a constant (C) can be added to the expected density function, and then, a distribution can be obtained via normalization. The proposed value of C can be the estimated proportion of the estimated smoothing kernel width (3 times its σ) and the code element width.

As the Gaussian kernel width increases, the code element contrast gets lower, and at some point, code elements become unreadable. This roughly happens when the kernel size exceeds the code element size. Taking this phenomenon into consideration, the maximum Gaussian kernels used in this thesis for the synthetic test set has the same width as the undistorted code element. The effect of noise is also added to the model with Gaussian distribution, having σ in the $[0, 0.25]$ range in the test database. Fig. 3.11(b) shows that these amounts of noise and blur theoretically



(a) QR-50 (b) QR-100

Figure 3.11: Enlarged QR code parts with different amounts of noise present.

destroy the value of a code element.

The value x in QR-x denote the percentage of noise and blur added, up to these discussed maxima (QR-0 and QR-100 are the perfect and the hardest cases of the data set, respectively). Visual codes having larger amount of noise and blur can be considered undetectable, thus making localization unnecessary.

The proper values of σ and C can be approximated empirically by showing test images to the camera, with solid cells of bright and dark intensities, and with lines of different thickness, and measuring the spread of intensity values in cell histograms of the acquired images.

When taking the amount of noise and blur into account according to the above model, the desired histogram to a particular camera setup can be expressed in the $[0, 1]$ interval as

$$U_{C,\sigma}(x) = C + (1 - C)\left(e^{-\frac{x^2}{(\varepsilon+\sigma)^2}} + e^{-\frac{(1-x)^2}{(\varepsilon+\sigma)^2}}\right), \tag{3.8}$$

where ε denotes a small positive value to prevent division by zero in the ideal case. After sampling the function and normalization of the data, a desired distribution is obtained. Different values of σ and C lead to different distributions (Fig. 3.12). As for cameras having low dynamic range, contrast stretching is recommended as a pre-processing step.

Additionally, histograms are more reliable when based on more pixels, however, the goal of this research was to use the least number of pixels possible. In order to overcome inaccuracies that come from histograms based on a small amount of pixels, histogram binning is recommended.

With a given number of histogram bins b the dissimilarity of the desired and mea-

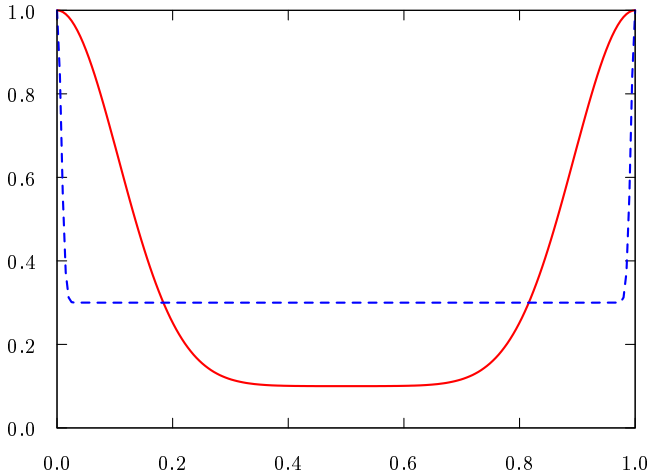


Figure 3.12: Expected probability functions. Red solid curve: small C (expected smoothing) with moderate amount of expected noise $C = 0.1$, $\sigma = 0.15$ (example for dirty environment), blue dashed curve: larger C with smaller amount of expected noise $C = 0.3$, $\sigma = 0.01$ (example for low quality phone camera).

sured histograms using various well-known formulas can be computed. Let $B(i)$ the number of pixels that fall into the i-th intensity bin in the measured cell histogram, $B_n(i)$ the normalized $B(i)$ and $U_n(i)$ the binned, normalized U-function

$$B_n(i) = \frac{B(i)}{\sum_{j=1}^{b} B(j)}, \tag{3.9}$$

$$U_n(i) = \frac{\int_{i/b}^{(i+1)/b} U_{C,\sigma}(x)}{\int_0^1 U_{C,\sigma}(x)}. \tag{3.10}$$

For simplicity's sake, the parameters C and σ are omitted from the notation of binned, normalized U-function and the derived distance measures. Nevertheless, for any choice of parameters C and σ, there exist a distance measure of the types defined below. Euclidean distance can be given as

$$D_e(B_n, U_n) = \frac{1}{2}\sqrt{\sum_{i=1}^{b} \left(B_n(i) - U_n(i)\right)^2}, \tag{3.11}$$

however, that is a bin-by-bin dissimilarity measure, which is sensitive to noise and the number of bins [69].

Normalized histograms can also be considered as probability density functions.

A common way to compare them is the Kolmogorov-Smirnov distance

$$D_k(B_n, U_n) = \max_i(|\hat{B}_n(i) - \hat{U}_n(i)|), \tag{3.12}$$

where $\hat{B}_n(i)$ and $\hat{U}_n(i)$ are cumulative histograms for the first i elements. This distance measure is widely used for cross-bin comparison of color histograms. Rubner et al. [59] proposed earth mover's distance (EMD) for comparison for multi-channel images, and normalized matching distance

$$D_m(B_n, U_n) = \frac{1}{b}\sum_i^b |\hat{B}_n(i) - \hat{U}_n(i)| \tag{3.13}$$

as a special case, which is suitable for grayscale histograms.

Table 3.1 shows similarity values using the aforementioned similarity measures ($S_X = 1 - D_X | X \in \{e, k, m\}$) for desired distributions $U_{0,0}$, $U_{0.12,0.12}$ and $U_{0.25,0.25}$, compared to synthetic QR codes of different levels of quality (Fig. 3.11), a flat histogram $F(x) = 1/b$, and histogram of a solid black image. I recommend using the matching distance D_m for histogram comparison, since it shows significantly higher values for this feature in the positive case.

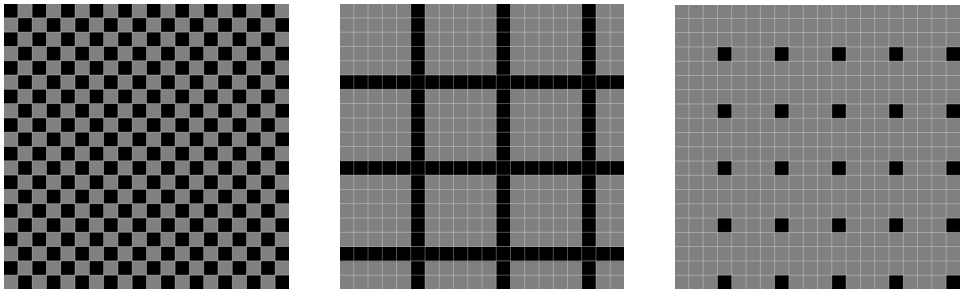
The results show that cell histograms of the hardest images of the set (QR-100) have cell histograms that show very small resemblance to the histogram of the model used in this thesis. This is due to noise, smoothing, and histogram asymmetry caused by the visual pattern variability of the embedded data. The consequence of this, is the increased false positive rate, or a limit to precision using higher thresholds for this measure.

For the making of cell histogram, not all pixels are necessary to be sampled from the cell, since any histogram that is made using properly selected partial pixel data can be considered an approximation of the one using all the pixels. Using only a fraction of the pixels is sufficient for images having only small amount of imperfections and noise, while it reduces the computation time. However, if the variance of QR code size is large, this under-sampling is quite risky because depending on the size of the code elements in the image, the proposed method might miss so many elements that the histogram looks very different from a typical QR code histogram.

Since the feature for recognition is based on histograms, using direction-independent or complex pattern is not a criterion, we only recommend uniform sampling of the cell. The checkerboard pattern is a good option, and the amount of pixels can be further reduced using other patters (Fig. 3.13).

Table 3.1: Euclidean, Kolmogorov-Smirnov and EMD similarity of measured histograms to the ideal histograms, using QR code pieces of different quality.

S_e	QR-0	QR-50	QR-100	Flat	Black
$U_{0,0}$	1.000	0.7653	0.6495	0.6938	0.6464
$U_{0.12,0.12}$	0.8381	0.9210	0.7950	0.8510	0.6111
$U_{0.25,0.25}$	0.7531	0.9587	0.8543	0.9243	0.5687
S_k	QR-0	QR-50	QR-100	Flat	Black
$U_{0,0}$	1.000	0.6739	0.5416	0.6250	0.5000
$U_{0.12,0.12}$	0.8045	0.8694	0.7196	0.8205	0.3045
$U_{0.25,0.25}$	0.7020	0.9327	0.7725	0.9024	0.2020
S_m	QR-0	QR-50	QR-100	Flat	Black
$U_{0,0}$	1.000	0.8599	0.7522	0.8125	0.5625
$U_{0.12,0.12}$	0.9140	0.9431	0.8382	0.8985	0.5625
$U_{0.25,0.25}$	0.8716	0.9610	0.8806	0.9409	0.5625



(a) checkerboard pattern (b) linear pattern (c) sparse pattern

Figure 3.13: Different cell patterns for histogram acquisition. Checkerboard pattern (a) using 50 %, 5:1 linear pattern using cca. 36 %, and sparse pattern using 6.25 % of pixels in cell.

For making a decision about cell size, two conditions have to be fulfilled. Cell size should be small enough to having many of them to build a QR code, so threshold would be chosen easily for dropping or keeping cell groups. On the other hand, cell size has to be large enough to provide reliable statistical data for the histogram, at least tens of pixels for each bin. The number of histogram bins can also be tuned to fulfill reliability and robustness. Too many histogram bins leads to sensitivity for noise, while choosing too few bins result in losing the feature. We recommend about 8 to 16 bins with 8-bit grayscale images.

The range of expected code size, the desired number of bins and the mean pixel count falling into each bin determines the usable pattern and block size for detection. From another point of view, the number of bins, the expected mean pixel count

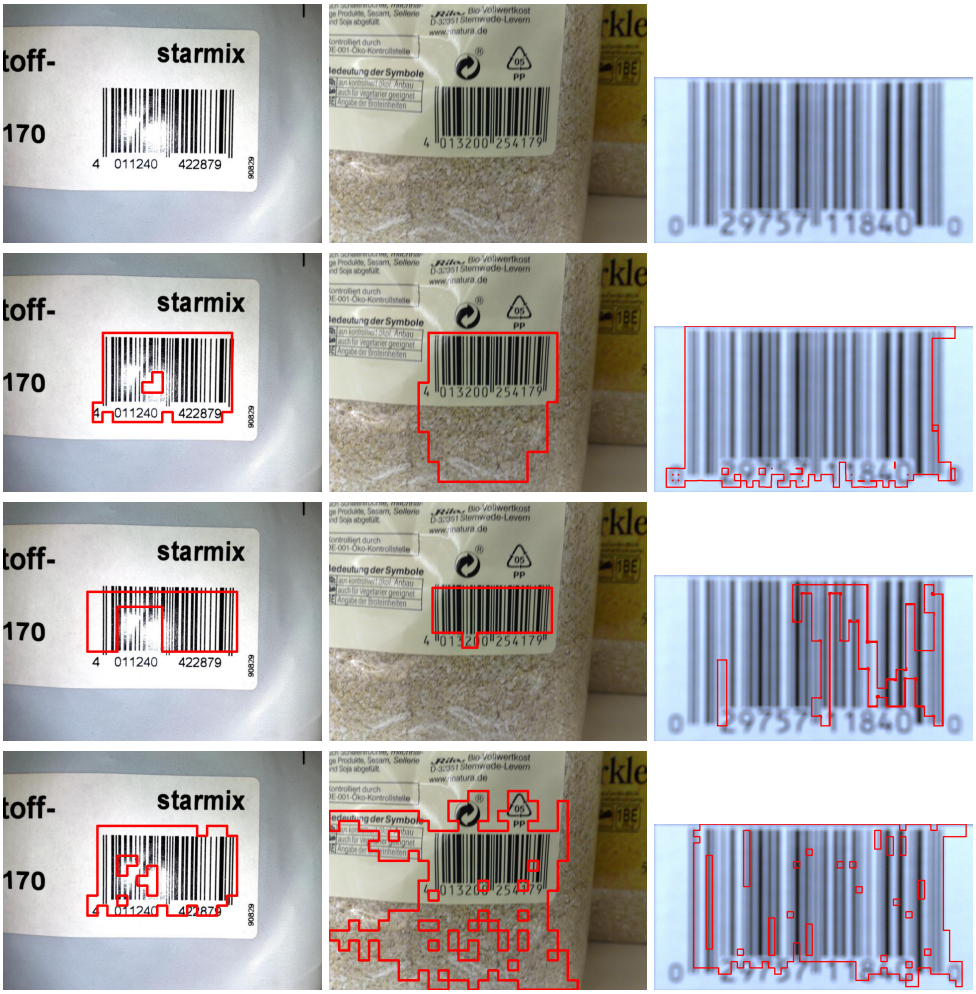


Figure 3.14: A qualitative test of barcode detectors based on our features. Contours of the possible barcodes are shown in red. From top to bottom: original image, distance transformation, local clustering, contrast measuring. Each column represents a different test image.

and the block coverage ratio of the chosen pattern defines the size of the smallest detectable visual code.

3.5 Evaluation and results

Similarly to Section 2.5, a qualitative comparison has been made to commercial softwares, and the results are shown in Fig. 3.14.

Table 3.2 shows test results of the algorithms involving image tessellation. Distance transformation and local clustering performed well, however, they showed a

Table 3.2: Average detection performance of the presented methods (expressed in percent).

Algorithm	Precision	Recall	Accuracy	F-measure
Tuinstra et al. [76]	57.08	85.29	84.19	48.39
Juett et al. [36]	34.26	94.08	72.76	36.13
Circular scanline	23.80	92.70	71.33	37.87
Distance trans.	20.00	95.85	54.52	23.54
Local clustering	81.68	72.34	89.22	62.12
Contrast measuring	51.17	88.02	82.58	49.07
HIST-FULL	83.20	93.82	97.32	88.19
HIST-CHK	85.76	90.35	97.37	87.99
HIST-LIN	66.25	93.89	94.26	77.68
HIST-SPA	67.29	90.14	94.29	77.05

moderate amount of false-positives. It is better to use distance transformation as a weak "classifier" instead of on its own. Even though, it produces the highest amount of false positives, it comes with high recall. It is more like an exclusion filter for image parts than a detector.

Four variants of the histogram-based algorithm was evaluated, each named after the pattern they use for building the cell histograms (HIST-FULL uses all pixels, HIST-CHK, HIST-LIN and HIST-SPA uses the checkerboard, linear and sparse patterns.

The effect of chosen threshold T to efficiency, using HIST-FULL, is shown in Fig. 3.15. Sensitivity drops below 1.0 at $T = 0.73$, and F-measure peaks at $T = 0.86$. For industrial setups, where localization of all codes is crucial, we recommend $T \approx 0.8$, since sensitivity is still 99 % and precision is about 50 %. The behavior for chosen threshold and noise level is similar in all chosen patterns. Fig. 3.16 shows that noise has no significant effect to false positive rate, it only drops sensitivity at higher rates.

The approach using a circular scan-line (Section 3.1.1) is very fast and has good recall rate, however, it produces significant amount of false positives, which the decoding algorithm has to handle.

Thanks to the nature of the Distance Transformation and Local Clustering methods, they are both regarded as reliable on images with minor distortions, unlike the Hough transformation, which detects barcodes based on line angles.

Partitioning the image, assigning a measure to each partition, and looking for homogeneous areas in the feature image is a general approach to detect patterns. With different features, like contrast variance, histogram information or distance

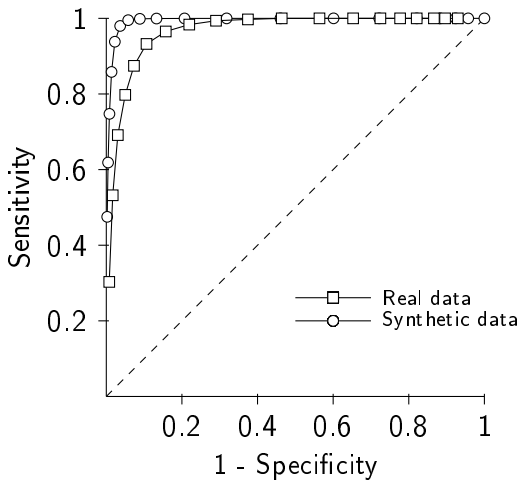


Figure 3.15: Efficiency of HIST-FULL algorithm according to different thresholds, visualized in ROC space.

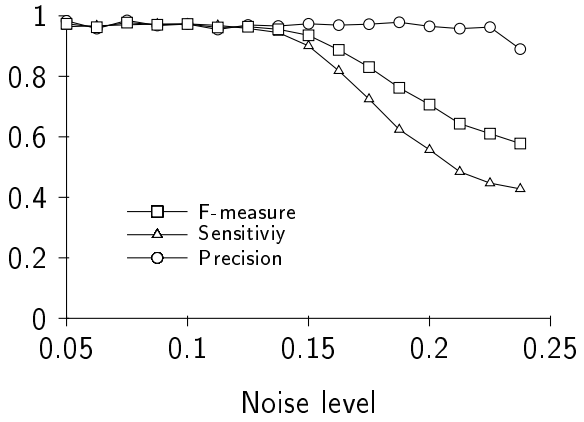


Figure 3.16: Precision, Sensitivity and F-measure of HIST-FULL according to noise level, using 0.86 as threshold

information, it can be used well as a barcode locator method.

3.6 Ensemble of simple features

Simple detectors can be aggregated in many ways, such as, majority voting, using the maximum value of all, or weighted voting [5]. Each approach is appropriate for fulfilling different goals. Majority voting can be applied with good results when the single detectors have relatively low precision with a moderate or high recall rate. In this thesis pixels with higher certainty are classified in this way, while

keeping false positives at low rate. Using the maximum value of the feature images produced by the individual detectors is good for maximizing the recall, for example, detecting all possible ROIs, but it dramatically decreases precision when the single detectors are weak on that property. However, we can use that approach in industrial setups, where detecting all barcode locations is of crucial importance. One can also experiment with weighted sum of the feature images. Since in this case, the intention was also to maximize recall, separate recall values of each simple detector based on only one feature were examined, after which, that value was used as their weight for the ensemble decision. This idea is summarized in Algorithm 1.

Algorithm 1 An ensemble of simple detectors.

$I = \text{Normalize\_levels}(I)$
$I_{canny} = \text{Canny\_edge\_detect}(I)$
$I_{sharp} = \text{Unsharp\_masking}(I)$
$I_{distance} = \text{Distance\_Transform}(I_{canny})$
$F_{minmax} = \text{Min\_Max}(I_{sharp})$
$F_{hough} = \text{Probabilistic\_Hough}(I_{canny})$
for t in *tiles* **do** ▷ Assign measures for the tiles
  $V_{distance}(t) = \text{Distance\_Transformation\_Measure}(I_{distance})$
  $V_{contrast}(t) = \text{Contrast\_Measure}(I_{sharp})$
  $V_{cluster}(t) = \text{Local\_Clustering\_Measure}(I_{sharp})$
end for
$\text{Filter}(V_{distance})$ ▷ Connected component labelling
$\text{Filter}(V_{contrast})$ ▷ and drop "small" blobs
$\text{Filter}(V_{cluster})$
for (x, y) in I **do** ▷ Build feature images from tile information
  $F_{distance}(x, y) = \text{GetTileValue}(V_{distance}, x, y)$
  $F_{contrast}(x, y) = \text{GetTileValue}(V_{contrast}, x, y)$
  $F_{cluster}(x, y) = \text{GetTileValue}(V_{cluster}, x, y)$
end for
$F_{majorvote}(x, y) = Maj(F_{minmax}, F_{hough}, F_{distance}, F_{contrast}, F_{cluster})$
$F_{maxval}(x, y) = Max(F_{minmax}, F_{hough}, F_{distance}, F_{contrast}, F_{cluster})$
$F_{weightedvote}(x, y) = 1/(\sum_{F_i \text{in} F} \text{recall}(F_i)) \times \sum_{F_i \text{in} F} F_i(x, y) \times \text{recall}(F_i)$
▷ F: the set of all feature images

The average performance indicators of the detectors are shown in Table 3.3. Detectors of this evaluation are based on Hough transformation [10], distance transformation [6], contrast measuring [10], local clustering [10], and MIN-MAX [10].

Table 3.3: Average detection performance of the proposed methods.

Algorithm	Precision	Recall	Accuracy	F-measure	Runtime
Hough trans.	64.83 %	85.07 %	84.22 %	58.76 %	230 ms
Distance trans.	20.00 %	95.85 %	54.52 %	23.54 %	190 ms
Local clustering	81.68 %	72.34 %	89.22 %	62.12 %	120 ms
MIN–MAX	26.39%	97.59 %	54.45 %	29.62 %	460 ms
Contrast measuring	51.17 %	88.02 %	82.58 %	49.07 %	140 ms
Majority voting	53.15%	85.70%	85.25%	48.44%	680 ms
Maximum of features	21.25%	96.45%	61.84%	24.51%	680 ms
Weighted voting	37.45%	91.97%	78.11%	37.35%	680 ms

3.7 Summary of the chapter

In this chapter, algorithms based on image tessellation were presented. The basic, global scan-line idea was extended to local scanning, using circular pattern [13]. Another variant was introduced as runlength measuring [10]. Other local approaches were evaluated as well, like cell histograms and distance transformation [6]. It was also shown that despite the fact that it is not recommended to use them individually, distance maps of the image cells can be used as a feature [6]. The final thought of this chapter indicated that even though features can be used in weighted unison [5], determination of importance for each feature is not trivial.

Chapter 4

Localization with neural networks

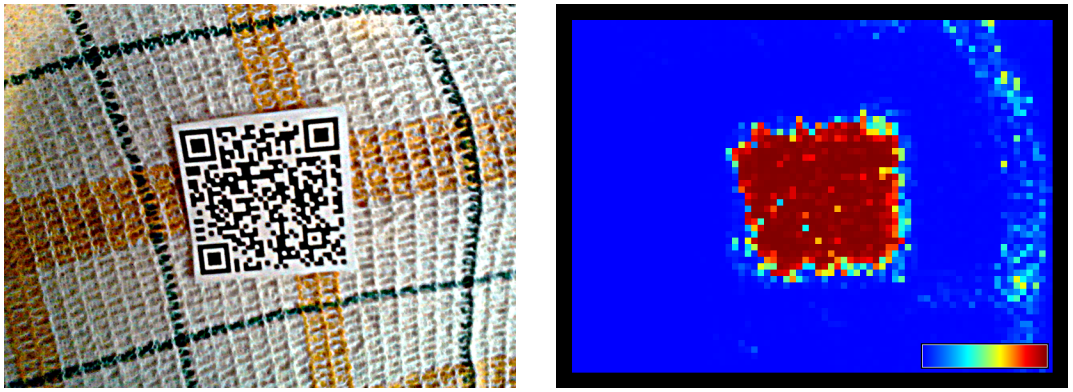
Both 1D and 2D visual codes have high variability regarding element layout. It would be problematic to manually enumerate all configurations or to construct features that give positive response to all of those configurations. However, with neural networks, learning the layout can be automatic and definition of patterns is not necessary.

4.1 Conventional and deep neural networks

During the last few years, there has been a renewed interest in applying neural networks, especially deep neural networks, for various tasks. As their name suggests, deep neural networks (DNN) differ from conventional ones (ANN) in that they consist of several hidden layers. However, if we want to train these deep networks properly, we have to be aware of the fact that the training method requires modifications as the conventional back-propagation algorithm encounters difficulties, like the so-called "vanishing gradient" and the "explaining away" effects.

In this very context the "vanishing gradient" effect means that the error might vanish as it gets propagated back through the hidden layers [29]. In this way some hidden layers, in particular those that are close to the input layer, may fail to learn during training. At the same time, in fully connected deep networks, the "explaining away" effects make inference extremely difficult in practice [34]. Several solutions have been proposed to counter these problems. The solutions modify either the training algorithm by extending it with a pre-training phase [34, 61], or the architecture of the neural networks [30].

In this chapter, it is concluded that deep neural networks are viable options for visual code localization.



(a) example real-life image (b) feature image

Figure 4.1: Image captured by a phone camera and visualized feature image according to the output of the neural network.

4.2 Image tessellation for neural network training

Similarly to the presented methods based on image tessellation (Chapter 3), the input vectors of the neural network are formed on block level. For each block, the neural network assigns a measure which reflects the probability of presence of a QR code part in that block, resulting in the feature matrix (feature image), that represents regions of interest (Fig. 4.1). The next step of the process is to find clusters in this matrix that have sufficient size, compactness and high values of probability to form a QR code. The final step is the same as the one for previously presented approaches. More precisely, the cluster centers that satisfy the above conditions are returned, and the bounding boxes for the QR code area candidates are given.

For each block the image is divided into, a vector is formed based on the information within the block. The extracted vectors are passed to the core of this approach, which is the neural network. Here, both conventional and Deep Rectifier Neural Networks (DRN) have been evaluated. DRNs alter the hidden neurons in the network and not the training algorithm by using rectified linear units. These rectified units differ from standard neurons only in their activation function, as they apply the rectifier function ($\max(0, x)$) instead of the sigmoid or hyperbolic tangent activation. Owing to their properties, the DRNs do not require any pre-training to achieve good results [30].

The rectifier function has two important properties, namely hard saturation at 0 and linear behaviour for positive input. The first property means that only a subset of neurons will be active in each hidden layer. For example, when we initialize the

weights uniformly, around half of the hidden units output are zeros. In theory, this hard saturation at 0 could harm optimization by blocking gradient back-propagation.

It shall be pointed out that the experimental results do not support this hypothesis, showing that the hard non-linearities do no harm as long as the gradient can propagate along some path [30]. On account of the other property of the rectified units, namely the linear behaviour of the active units, there is no "vanishing gradient" effect [30]. This linear behavior also means that the computational cost will be smaller, as there is no need to compute the exponential function during the activation calculation, and the sparsity can also be exploited.

There is also a disadvantage because of this linearity, called the "exploding gradient" effect, which means that the gradients can grow without limit. To prevent this, L1 normalization is applied by scaling the weights such that the L1 norm of each layer's weights remained the same as it was after initialization. What makes this possible is that the subset of active neurons behaves linearly for a given input, so a scaling of the weights is equivalent to a scaling of the activation.

Deep networks used in this thesis consisted of three hidden layers and each hidden layer had 1000 rectified neurons, as a DRN with this structure yielded the best results on the development sets. The shallow neural net was a sigmoid net with only one hidden layer, with the same number of hidden neurons (3000) as that for the deep one.

The output layer of the neural networks consisted of two softmax neurons (one for the positive label and one for the negative label), allowing the networks to output not only classification decisions but also posterior probability values. As for the error function the cross-entropy function was applied.

Two regularization methods were utilized to prevent overfitting, namely early stopping and weight decay. Early stopping was achieved by stopping the training when there was no improvement in two subsequent iterations on the validation set. As for the weight decay regularization, the weights were scaled back after each iteration, forcing them to converge to smaller absolute values than they otherwise would do.

The neural networks were trained using semi-batch back-propagation, the batch size was 100. The initial learn rate was set to 0.001 and held fixed while the error on the development set kept decreasing. Afterwards, if the error rate did not decrease for a given iteration, the learn rate was subsequently halved.

Figure 4.2: Example QR code block read using a circular pattern, with its corresponding intensity profile, a positive input vector for the neural network.

4.2.1 Input vector selection

For input vectors, a couple of options are available. Neural networks can work with partial or full data, and they can learn on samples of both the image and frequency domains. Moreover, in the frequency domain, quantized data can also be chosen, if no raw vectors are available.

For each block the image is divided into, a one-dimensional vector is formed by reading pixels in a circular pattern as proposed in Section 3.1.1, which can be adapted to QR codes (Fig. 4.2). This pattern provides prominent features while keeping the number of required pixels low, typically only 6-10 % of the image pixels. Moreover, the circular pattern can indicate the presence of a QR code in any orientation.

The training vectors are labeled positive if the QR code coverage ratio was higher than a selected T_c threshold for that block. Typically, the F-measure peaks are at $T_c \approx 0.5$, while $T_c \approx 0.1$ leads to better recall (hit rate). However, the number of these partially covered blocks is one order of magnitude smaller than the one of empty and fully covered blocks, hence T_c cannot be determined during the training phase. Furthermore, even if the DRN misses partially covered blocks, it only means it misses the perimeter of the code object. The expansion of the positively classified cell groups of the feature matrix overcomes this problem.

When making a decision about the block size, the same conditions apply like in

the case of cell histograms (Section 3.4). The block size has to be sufficiently large to supply prominent characteristics in the input vectors of the neural network.

The amount of overlap has to be also determined. Different offsets for blocks having the optimal size are also evaluated empirically, because papers on this topic do not categorically state whether overlapping improves neural network performance. However, convolutional neural networks are widely used, which motivates evaluation with different block offsets.

4.2.2 DRN training with JPEG DCT blocks

JPEG [78] is one of the most common still image formats, and provides efficient data storage and transfer. Most cameras can acquire images directly to JPEG format, and some devices can even output a stream of JPEG images, which motivates research into image processing methods using this format.

Neural networks are also capable of learning in the frequency domain, and JPEG format can be handled as a subset of that domain, also having fixed 8×8 px block size for input. For this case, one of the deep rectifier networks proposed in this thesis works directly with the DCT coefficients of the JPEG image. Using this approach, only the first steps of the decompression have to be performed for code localization, while the most complex step, inverse DCT can be skipped.

For setups using JPEG images or streams, the JPEG decoding process can be halted at the point where quantized DCT coefficients are restored from the file, right after the execution of the inverse of RLE and Huffman-coding. The matrix of coefficients that represent a 8× 8 px block in the image, serves as the input vector of the DNN. For the order of the coefficients, the "zigzag" pattern is appropriate as it is defined by the JPEG standard, since there is no difference in learning efficiency when using differently ordered vectors of the same training set. This order also correlates with the visual importance level of the coefficients, and hence it is recommended for the evaluation of the DNN performance using only a prefix of the vector.

The input vectors for this specific DRN are one-dimensional vectors, formed by the quantized DCT coefficients of a 8×8 px block. During the decoding, the multiplication with the quantization table can be omitted for two reasons. Firstly, due to the nonlinear nature of neural networks, they are capable of learning on a vector set and on the same set multiplied element-wise with another fixed vector, with similar efficiency. Secondly, the components of the input vectors were normalized as described in [41] to have zero mean and unit variance. This normalization improved the numerical condition of the optimization problem during training, ensuring a

faster convergence.

With this setup, the DRN has to be trained using images of the same compression level as images of the end-user application, since the original DCT matrix was not applied, which would require a multiplication by the quantization table. Without de-quantization, different levels of compression applied to the same image content lead to different vectors. These vectors can be far away from the training samples of a specific compression level. To overcome this, DRNs can be trained using the de-quantized coefficient vectors, which are roughly the same at similar compression levels.

4.3 Evaluation and results

For the input data, various options were available and a separate training was performed with each input type (Table 4.1). The first choice was to train a neural network on raw pixel data, and read in the chosen pattern for each block. The binary version of the vectors was also evaluated to compare the efficiency of the neural networks on grayscale and binary images. Also, since QR code has a well-defined, strict structure, it means that blocks of QR code parts probably have very specific components in the frequency domain. This assumption served as a motivation for performing to experiments with vectors in that domain.

The training vectors were transformed to the DCT and DFT domains, and neural networks were evaluated on both sets. In this case, my interest was specifically focused on the DRN performance in the DCT domain, since many setups involve JPEG compression by hardware, which can also be used.

Also, a neural network was trained on the edge map, since the structure of QR codes also suggest very specific edge layouts. Training vectors in that case consisted of pixels of the unified magnitude map of Sobel-X and Sobel-Y gradients (Fig. 4.3), read in a circular pattern.

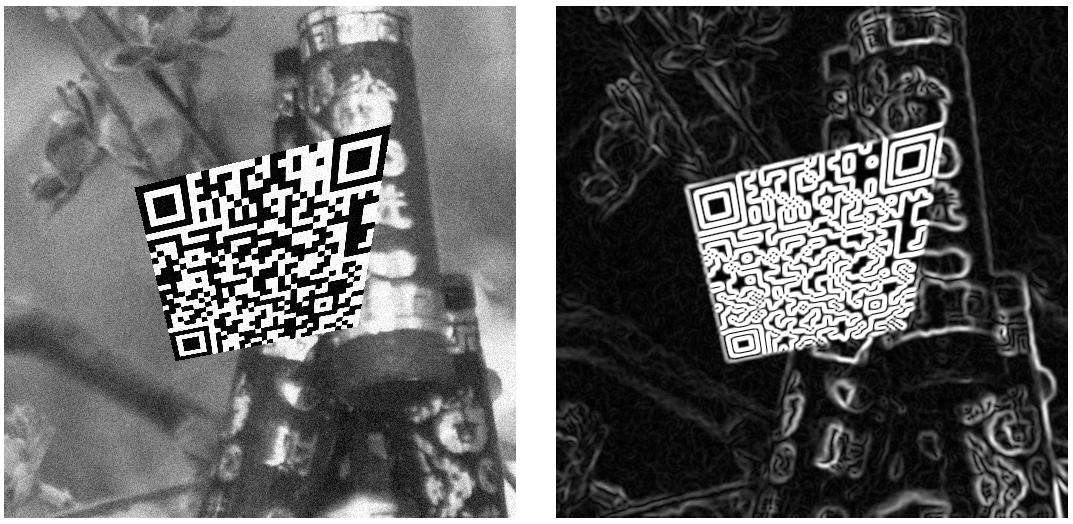
Results show that 8-bit input vectors are only slightly better than binary versions. The frequency domain and the edge map are both suitable for input, however, they are not worth the computational cost of the transformation, if the input is not in those format by default.

4.3.1 JPEG quality and number of coefficients

The effect of JPEG compression on DRN performance was also examined in this thesis. As it was expected, a better image quality resulted in better training results,

Table 4.1: Training results for different input data types and ranges.

Input data	Range	Precision	Recall	F-measure
ANN				
Raw pixels	8-bit	0.9892	0.9953	0.9922
Raw pixels	binary	0.9705	0.9846	0.9774
DCT	8-bit	0.9889	0.9951	0.9920
DCT	binary	0.9693	0.9860	0.9776
DFT	8-bit	0.9904	0.9981	0.9943
DFT	binary	0.9711	0.9808	0.9760
Sobel-XY	8-bit	0.9979	0.9990	0.9984
DRN				
Raw pixels	8-bit	0.9947	0.9972	0.9959
Raw pixels	binary	0.9704	0.9862	0.9782
DCT	8-bit	0.9941	0.9967	0.9954
DCT	binary	0.9686	0.9873	0.9778
DFT	8-bit	0.9933	0.9958	0.9945
DFT	binary	0.9621	0.9850	0.9734
Sobel-XY	8-bit	0.9978	0.9991	0.9984



(a) example synthetic image (b) edge magnitude map

Figure 4.3: A sample synthetic image and Sobel magnitude image. Edge maps are also reliable inputs for the neural network training.

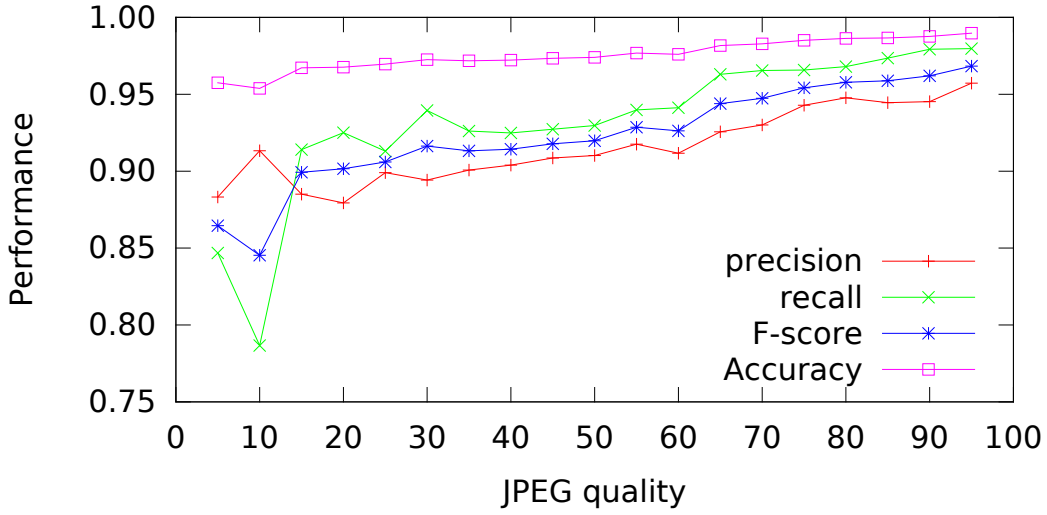


Figure 4.4: DRN performance for the quality of the data set.

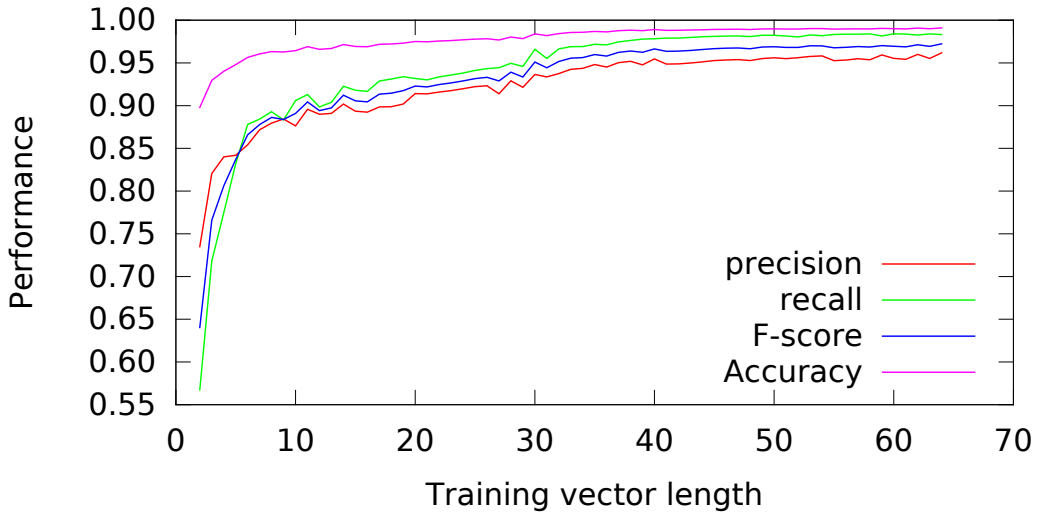


Figure 4.5: DRN performance for the length of the training vectors.

as shown in Fig. 4.4.

Fig. 4.5 shows the performance measures of DRNs trained on the first n elements of the vectors. It can be observed that roughly the first 10 elements of the coefficient vector are sufficient for training a DRN that has an F-measure above 0.9, and the performance only slightly improves when using more than half of the vector coefficients.

All the networks trained with a test set of specific quality were also evaluated on all the sets, hence showing the robustness of the networks relative to the difference

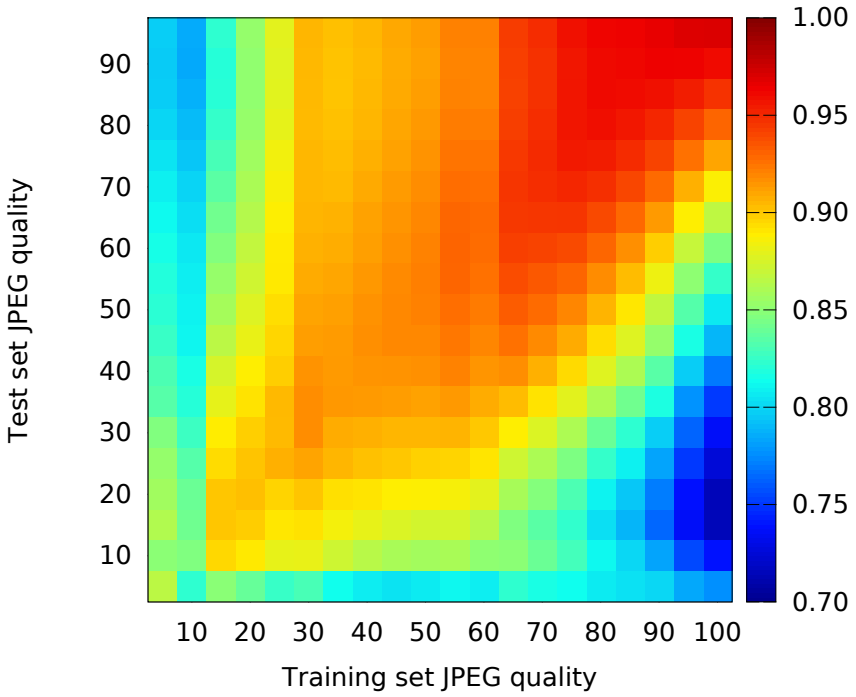


Figure 4.6: Performance map of the trained DRNs used on images having the same content but different JPEG quality.

in compression levels between the training and test sets (Fig. 4.6). The results show that DRNs are quite specific to the quality they were trained on, but still perform well up to a 10–15 % tolerance. Based on these results, although the training of a DRN to a specific image quality is not required, they perform best when training and testing images are of the exact same quality. An analogous rule applies when training the DRN using all the vectors extracted from images of various JPEG quality. The DRN trained using those vectors would perform worse compared to the one trained with the same specific JPEG quality as the end-user setup.

4.3.2 Partially covered blocks

In the first case, training was only performed using the vectors for the background and blocks fully covered with QR code parts. As the next step of this experiment,

Table 4.2: Evaluation of the neural networks also having vectors of partially covered blocks as input, with different thresholds for input labels.

NN type	Data type	Precision	Recall	F-measure
$T^+ = 0.1$				
ANN	Real	0.6454	0.9518	0.7692
DRN	Real	0.6699	0.9419	0.7829
ANN	Synthetic	0.5654	0.9901	0.7198
DRN	Synthetic	0.5630	0.9865	0.7169
$T^+ = 0.5$				
ANN	Real	0.9175	0.5994	0.7251
DRN	Real	0.8962	0.8414	0.8679
ANN	Synthetic	0.8703	0.8947	0.8823
DRN	Synthetic	0.9059	0.9347	0.9201

Table 4.3: DRN results on different subsets of input vectors of synthetic images. N^+ and N^- denote positive and negative samples, while N_b^+ and N_b^- are subsets of N^+ and N^- with partially covered blocks excluded.

Subset	Opt. Thresh.	max($F1$)	AUC
All vectors	0.62	0.9343	0.9957
$N^+ \approx N^-$	0.63	0.9270	0.9949
$N_b^+ \approx N_b^-$	0.82	0.8312	0.9608

the input data was extended for partially covered blocks, and thresholded at 0.1 and 0.5 coverage ratios for positive labeling. No filtering was applied to the amount of negatively labeled vectors. Both ANNs and DRNs were trained separately on vectors of both synthetic and real images.

The results (Table 4.2) indicate that allowing the partially covered samples for the training significantly reduces the precision of the NNs. Furthermore, a threshold of 0.1 for the positive labels keeps the recall at a satisfactory level, while 0.5 decreases it dramatically (Fig. 4.7). Still, the low recall is not a problem, since enough positively predicted blocks remain in the probability matrix to be able to indicate a QR code candidate even in these cases (Fig. 4.7(e)).

Consequently, it may be concluded that NNs trained on full images and on a reduced subset of vectors (where positively and negatively labeled samples are roughly the same amount) do not differ significantly, while excluding vectors that come from partially covered blocks dramatically reduces the performance. Table 4.3 summarizes the results obtained by using the aforementioned training sets.

Table 4.4: Efficiency of neural networks trained for regression.

NN type	Data type	MSE	AUC	R^2
ANN	Real	0.0953	0.5416	0.0794
DRN	Real	0.0490	0.9837	0.1960
ANN	Synthetic	0.0537	0.9682	0.2874
DRN	Synthetic	0.0465	0.9782	0.2918

4.3.3 Regression training

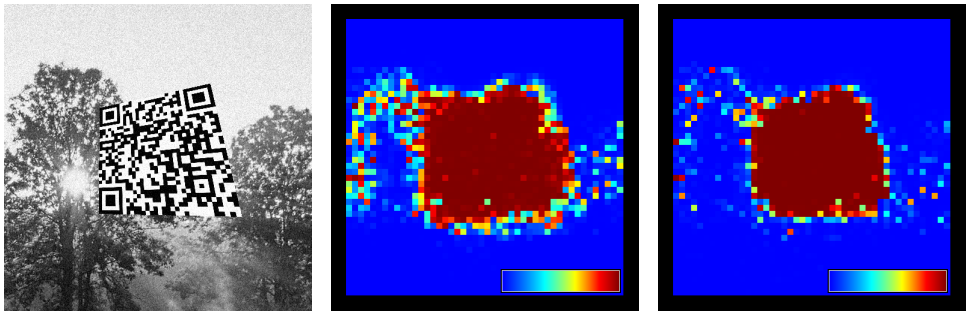
Instead of training a classifier neural network for the partially covered blocks, networks can also be trained for regression using the MSE error function and one sigmoid output neuron. Table 4.4 shows the results of the training. Both ANNs and DRNs were evaluated on synthetic and real test data using all the blocks of images, which means about 4 times more such input vectors that contain no QR code parts compared to the amount of those that do. Common quality measures of regressions, like MSE, AUC and R^2, were used to indicate the goodness of fit. The feature images of the regression can be seen in Fig. 4.8.

It can be clearly seen that the amount of vectors extracted from real images were insufficient for the training. Good AUC values suggest that by using the right threshold we can achieve good results with DRNs. Unfortunately, regression training fails to improve the results compared to those obtained with the standard classification training.

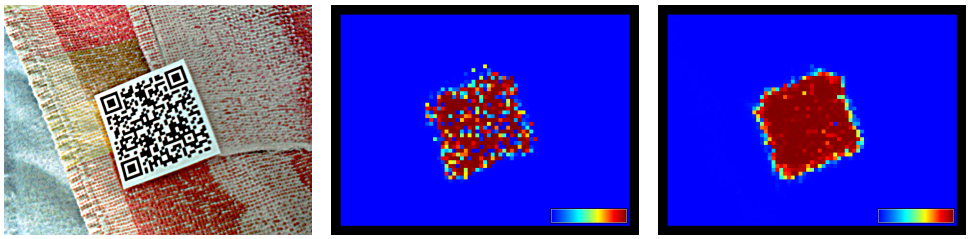
4.3.4 Domain adaptation

Neural networks need a lot of training vectors which might not be available for each end-user application. Artificially generated training vectors extended with ones of the final setup can improve precision, as shown in Fig. 4.9. Networks were trained using the 1.5 million vectors of the synthetic database, which had low precision on the real data set, which could be improved by adding vectors of the latter set.

In the final adaptation step, when we used all 1.75 million training vectors, our DRN achieved an F-measure of 86.81 on the real data. This performance is slightly better than the one achieved by training a DRN only on the real data, but requires more training time.

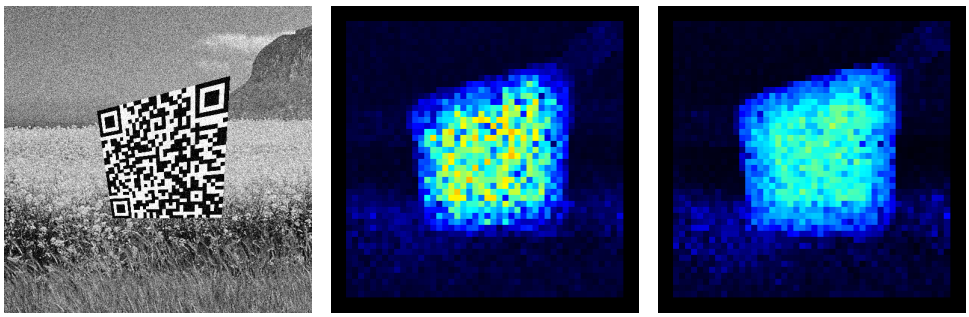


(a) Synthetic image (b) ANN feature image (c) DRN feature image

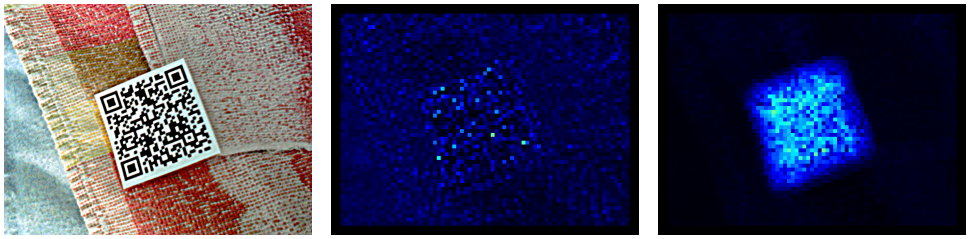


(d) Real image (e) ANN feature image (f) DRN feature image

Figure 4.7: Original and feature images built from the output of the neural networks.



(a) Synthetic image (b) ANN feature image (c) DRN feature image



(d) Real image (e) ANN feature image (f) DRN feature image

Figure 4.8: Original and feature images for the regression training, blue-cyan-orange linear heat map. The 4:1 ratio of vectors having no code part pushes the predicted values to a lower range than those for the previous case.

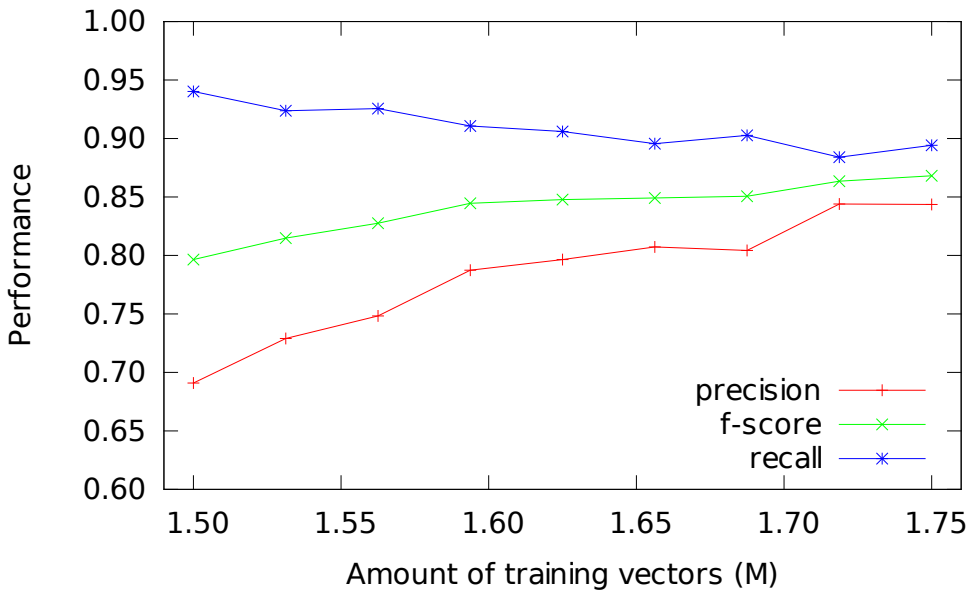


Figure 4.9: Domain adaptation capabilities. We gradually added 0.25 million vectors from the real database to the 1.5 million vectors of the synthetic database.

4.3.5 Other patterns and code types

While the circular pattern is shown to be appropriate for the block evaluation, various other patterns were also evaluated (Table 4.5). When only every nth pixel is read along the axes, the F-measure drops from 0.95 ($n = 1$) to 0.89 ($n = 10$). Reading the pixels along block center-lines or diagonals is also effective and involves only a few pixels. In this particular case, histogram learning turns out to be the least reliable method.

Regarding different code types (Table 4.6), learning effectiveness does not vary significantly, except in the case of PDF417 codes, which is not a real 2D code type but a stacked 1D barcode, having the smallest rotation-invariance.

4.3.6 Parameters and running times

For neural networks, a custom setup was used, which was implemented to run on a GPU [75]. The training of a DRN on the synthetic dataset took less than 8 minutes using an NVIDIA GTX-770 graphics card. The computation power of this setup allowed us to process about 450 000 vectors per second, which means a real-time processing of 800×600 px images with about 5 FPS using 60 px for the block size and 10 px for the offset, and 15 FPS for images with JPEG DCT blocks.

Table 4.5: Evaluation on different patterns. The pattern-n denotes for a pattern where every nth pixel is read along each axis. Pattern-X contains only the pixels of the two diagonals, and Pattern-Plus the ones of the horizontal and vertical centerlines of the block. Pattern-Hist is the block histogram.

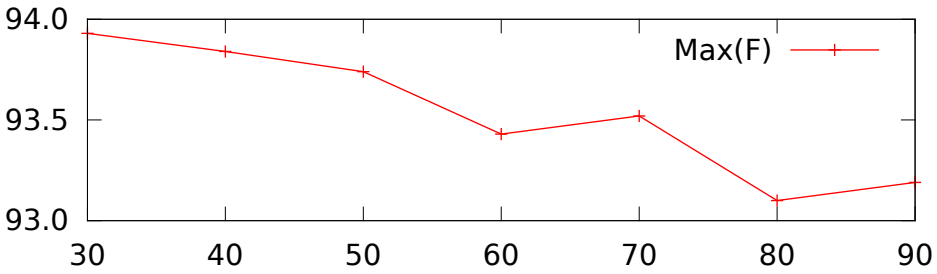
Input	Precision	Recall	F-measure
Pattern-1	0.9465	0.9688	0.9575
Pattern-2	0.9404	0.9630	0.9516
Pattern-3	0.9359	0.9654	0.9504
Pattern-4	0.9322	0.9567	0.9443
Pattern-5	0.9162	0.9539	0.9346
Pattern-6	0.9263	0.9157	0.9210
Pattern-7	0.9203	0.8884	0.9041
Pattern-8	0.9254	0.8825	0.9034
Pattern-9	0.9015	0.8975	0.8995
Pattern-10	0.8925	0.8898	0.8911
Pattern-Plus	0.9320	0.9682	0.9497
Pattern-X	0.9200	0.9334	0.9267
Pattern-Hist	0.8644	0.8905	0.8773

Table 4.6: Evaluation on popular, two-dimensional code types.

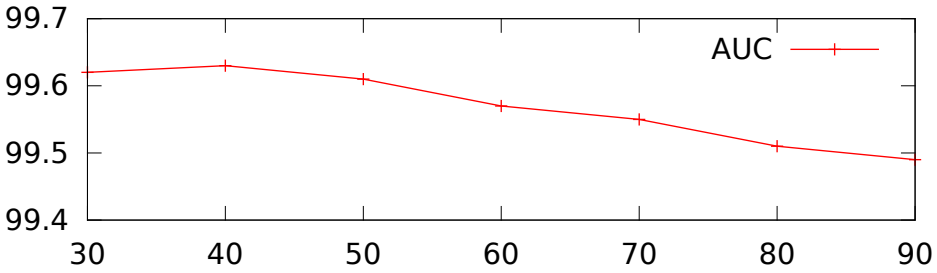
Type	Precision	Recall	F-measure
QR	0.9879	1.0000	0.9939
PDF417	0.8888	0.9032	0.896
Data matrix	0.9780	0.9888	0.9834
Codablock	0.9764	0.9431	0.9595
Aztec code	0.9906	0.9883	0.9894

The QR codes used in the evaluation had a 10 px element size. Block sizes of the input vectors from 30 px to 90 px were evaluated. Block sizes smaller than 30 px would lead the training to learn solid black and white blocks as positive, thus significantly raising false positive rate. Block sizes larger than 90 px would also decrease the performance, since a large block size drastically decreases the number of positive samples available for the training while introducing a big amount of partially covered blocks that are harder to learn. The values in Fig. 4.10 also tell us that the proposed method is robust for expected QR code size, since the block size range for good performance is sufficiently large according to code dimensions.

Fig. 4.11 shows the results got for DRNs using different rates of overlapping up to 60 px (no overlap). It is notable that the F-measure differs only slightly for overlap sizes from 10 px to 60 px, while the amount of input vectors is dramatically



(a) max(F)



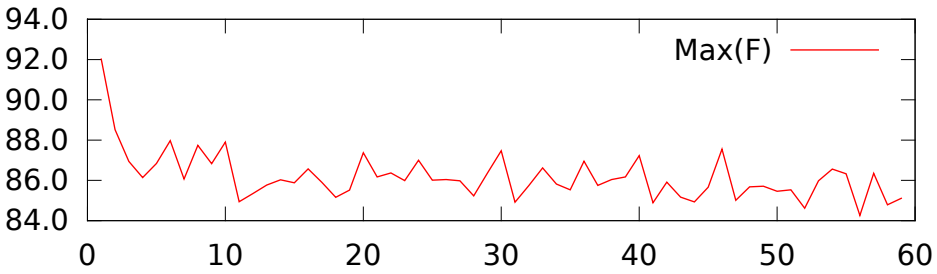
(b) AUC

Figure 4.10: DRN results respect to input block size.

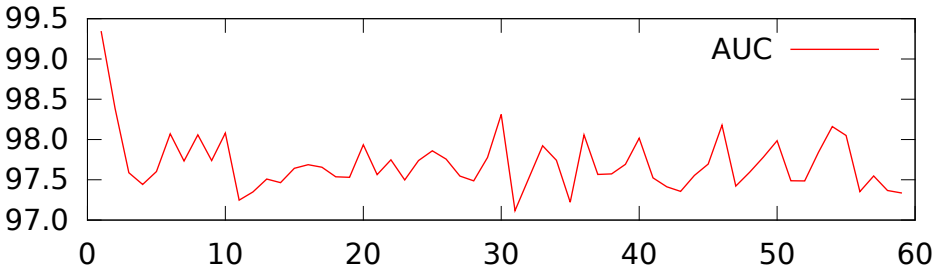
smaller. For JPEG DCT blocks, we only used the original 8×8 px block size with no overlap.

After carrying out the experiments, a DRN was trained with the best experimental training setup, in order to compare it with other localization methods outlined in the literature. According to Table 4.7, NN training is shown to be a viable option for real-time efficient QR code localization. As the first approach for comparison purposes, an algorithm based on mathematical morphology was selected [52], since it is also a general purpose toolset of image processing, as NNs are meant for machine learning tasks. This reference method is reliable, but due to the morphological operations, it is computationally intensive and has the slowest processing speed (from 900 to 1300 ms per image). Another study on QR code localization uses specific, image-based classifier training (see Chapter 5). It proposes training a cascade of classifiers using Haar-like features [3].

Precision of the DRN on the real set is worse, which is probably due to the smaller number of training vectors used. Also, the QR codes were slightly larger compared to the training set, which resulted in less prominent patterns to learn within an 8×8 px block. High-resolution images having QR code structure elements larger than the JPEG block size are required to be downsampled before processing.



(a) max(F)



(b) AUC

Figure 4.11: DRN results for input block offset using a 60 px block size.

Table 4.7: Performance comparison of the best DNN with other algorithms, REF-HAAR [3], REF-MORPH [52], REF-OHBUCHI [19] and REF-LIN [46].

Algorithm	Precision	Recall	F-measure
DRN RAW	0.9360	0.9327	0.9343
DRN JPEG DCT	0.9607	0.9770	0.9899
REF-HAAR	0.1535	0.9436	0.2640
REF-MORPH	0.6989	0.9930	0.8042
REF-OHBUCHI	1.0000	0.8370	0.9113
REF-LIN	0.9340	0.9490	0.9414

This also has a positive effect on the processing speed, as it greatly reduces the number of blocks that have to be evaluated. The downsampled image computation can be avoided by setting the camera to acquire lower resolution images.

DRNs have to be trained using vectors that come from QR codes of comparable size to the expected test images. This limitation can be overcome using multiple DRNs trained on vectors of various QR code sizes, or one single DRN trained on all expected sizes, which will lead to a worse overall performance than that for given particular neural networks.

Performance measures were computed by treating each block separately. How-

ever, groups of blocks with appropriate size and shape have to be extracted from this feature map by clustering or connected component labeling, and bounding boxes have to be defined in order to send cropped image regions data to decoding algorithms [71, 79].

4.4 Summary of the chapter

Neural networks, especially deep rectifier networks both give viable alternatives for visual code localization. Having sufficient number of training vectors, they are capable to efficiently classify image cells whether or not they contain code segment. They work for various inputs, from raw pixel data to the 1D vector formed by the circular pattern introduced. It is remarkable that deep rectifier networks are capable to work on input vectors of the DCT domain, which enables them to process JPEG images with minimal computational cost [4]. Results of this chapter were published in [31] and in [13].

Chapter 5

Cascade classifiers using simple features

Classification is a decision to identify different groups of elements (classes), based on their attributes. The decision about elements of unknown class is usually based on learning, which is the process of forming a knowledge base from elements of known class. In image processing, classification is mostly identification of objects or textures, based on attentive geometric, or more complex features derived from pixel data [64]. Image segmentation has similar objective than classification.

Methods based on wavelet transformation [7, 54] look at images for barcode-like appearance by a cascaded set of weak classifiers. Each classifier working in the wavelet domain narrows down the possible set of barcodes, decreasing the number of false positives while trying to keep the best possible accuracy.

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, 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 preceding 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 features.

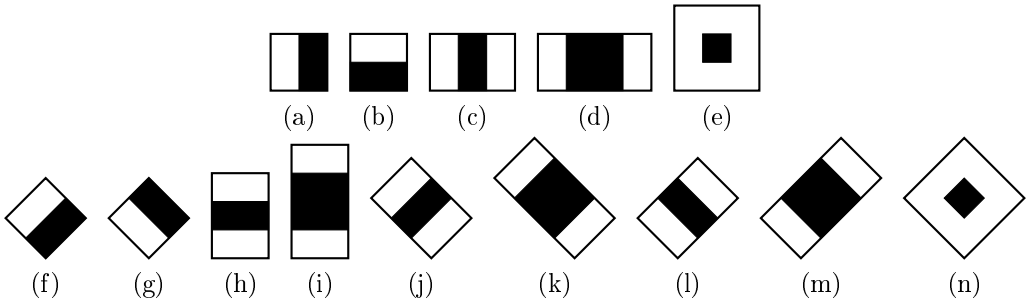


Figure 5.1: Haar-like features: edge type (a, b), line type (c, d), center type (e), and rotated entities (f–n).

5.1 Feature selection

In image processing, Viola and Jones [77] introduced the use of Haar-like features as the core of these weak classifiers. There are three sets of features, edge-type, line-type and center-type, and each set has its 45-degree rotated extension (Fig. 5.1), proposed by Lienhart et al. [44].

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 scale, 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, Gentle AdaBoost was used in order to increase accuracy.

Feature evaluation time can be further reduced by using integral images (Fig. 5.2) for the evaluation, which is derived from the original image as

$$I_{\text{int}}(x,y) = \sum_{u<x,v<y} I_{\text{orig}}(u,v), \tag{5.1}$$

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

$$\sum_{\substack{A_x \le x < C_x \\ A_y \le y < C_y}} I_{\text{orig}}(x,y) = I_{\text{int}}(C_x,C_y) - I_{\text{int}}(B_x,B_y) - I_{\text{int}}(D_x,D_y) + I_{\text{int}}(A_x,A_y), \tag{5.2}$$

where A, B, C and D denote for the corner points of the rectangle (Fig. 5.2).

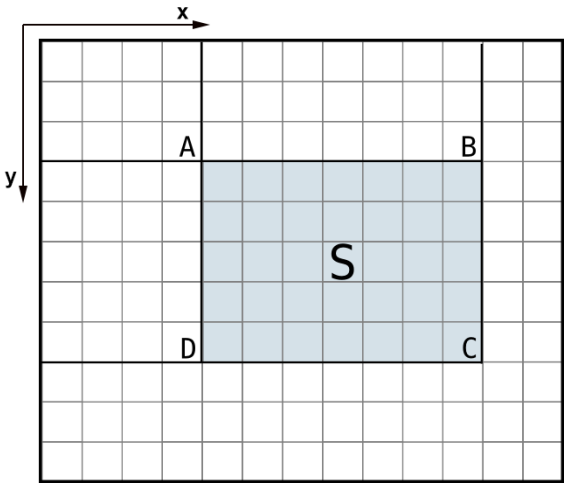


Figure 5.2: Calculation of the intensity sum over rectangle S. Intensity sum of S is calculated as $\mathrm{Rect}(O,C) - \mathrm{Rect}(O,B) - \mathrm{Rect}(O,D) + \mathrm{Rect}(O,A), O = \{0,0\}$ Integral image already has intensity sums calculated from O to points A, B, C, D, thus simplifying intensity sum query to $\mathcal{O}(1)$ with respect to rectangle size.

Belussi et al. [3] experimented with classifiers based on Haar-like feautres, and proposed parameters for visual code localization. Their classifier was trained on the Finder Pattern (FIP) of the code object. According to their experiments, the most accurate classifier uses only the basic set of wavelets, weak classifiers are organized into cascade topology with a maximum 1 split, each having 50 % false positive rate and 4000 samples of 16×16 px size.

Instead of Haar-like features, Local Binary Patterns (LBP) and Histograms of Oriented Gradients (HOG) can also be used for the feature evaluation. The concept of partitioning of the image, and reading each block in a circular pattern (Section 3.1.1) is analogous to LBP [53], with the main difference of not using the center point for making the feature.

HOG descriptors were first introduced in pedestrian detection [20], however, they are often used in areas of computer vision where LBP, SIFT or shape context is applicable. There are some special cases [80] where LBP and HOG can be used together with improved overall accuracy, too.

5.2 Input data

While FIPs have well-defined structure (Section 1.1.2), it is also proposed in this thesis to train classifiers on the whole code area. That way, more features can fit, and

post-processing is simplified significantly. Moreover, I also propose improvements for the post-processing step of FIP-training.

5.2.1 FIP training

A classifier based on a Haar-like feature set is already discussed in the literature [3], and will serve as a reference method to 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.

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 a smaller portion for training 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 they can be Classification And Regression Trees (CART) themselves, with a given number of maximum splits.

Classifiers based on simple features are boosted by Gentle AdaBoost. To avoid combinatorical explosion of the parameter tuning, Belussi et al. [3] experimented

with individual parameter variation while keeping other parameters fixed, and documented their empirically observed optimal parameter values for the Haar-training on FIPs.

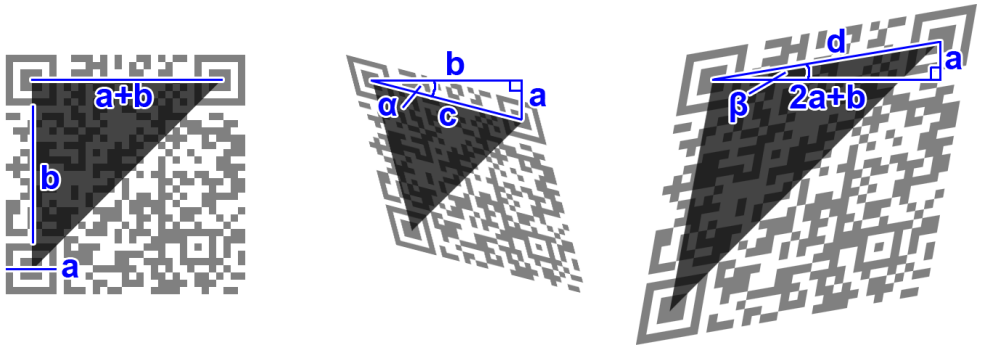
The number of stages were set to 10, according to the experiments of Belussi et al. [3]. For the first four stages, using only one feature was sufficient to reach the TPR and FNR defined above, while in later stages, more features were required, from 9 up to 15. The training did not contain a priori information about which features to prefer, they were chosen empirically as it is implemented in the OpenCV library.

5.2.2 Triplet formation for FIP-based classifiers

For the classifiers trained to FIPs, post-processing is needed to reduce the amount of false detection. Belussi et al. proposes searching through the set of FIP candidates 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. Assumptions had to be formed 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. 5.3(a)).

To the synthetic image set, perspective distortions were added, that were capable of shifting the FIPs of the QR code by one FIP width at most. 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. 5.3(a)). A QR code having a distortion that warps the FIP center inward by a (Fig. 5.3(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. 5.3(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



(a) (b) (c)

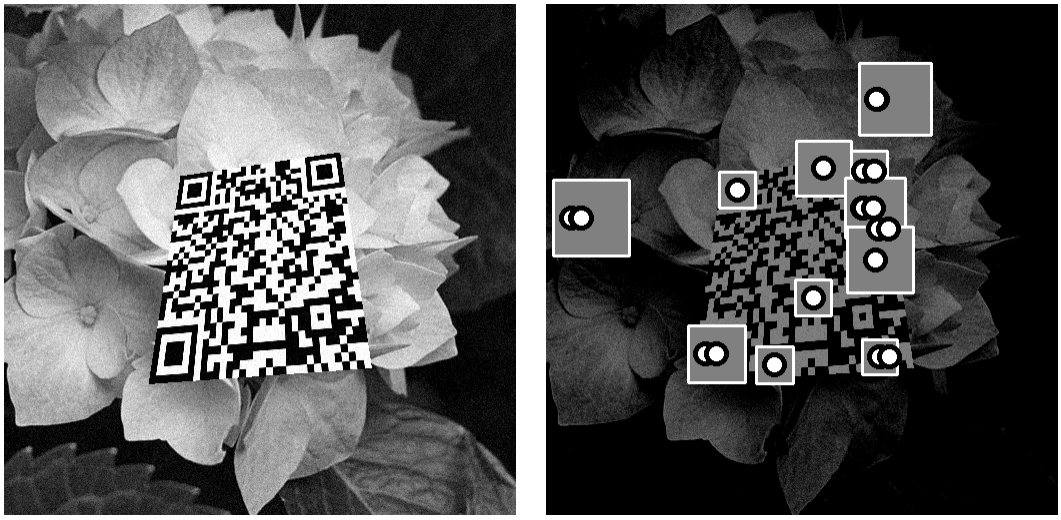
Figure 5.3: Example for deciding on triplet formation tolerance. From the top left corner of a perfect QR code, the other two FIPs are enclosing 90 degrees and distance of FIP centers is $a+b$ (a); Considering two cases of distortion where FIPs are shifted inwards (b) and outwards (c) with a, distances and angle tolerances for acceptance can be calculated using basic geometry.

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.

Belussi et al. [3] 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.

FIP-trained classifiers 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. 5.4).

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

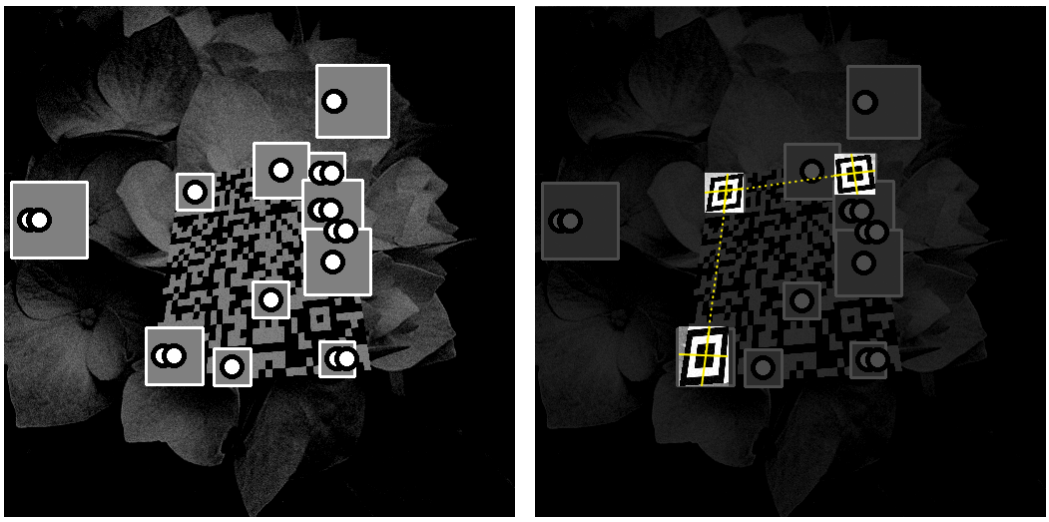


(a) (b)

Figure 5.4: 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. 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. 5.5).



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

Figure 5.5: Scan-line post-processing of the FIP candidates in order to reduce false positives.

5.2.3 Full code area training

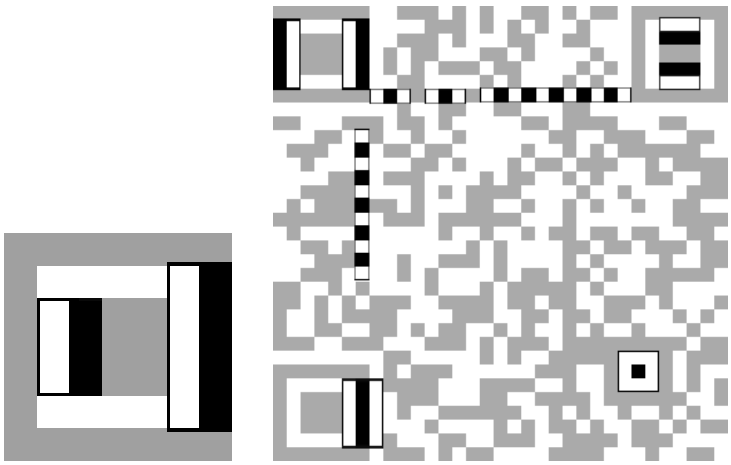
In order to increase the strong features of the object intended to detect, this thesis proposes training of a classifier for the whole code area. Even though QR codes have high variability on the data region, they contain data density patterns, a fourth, smaller FIP that can be perfectly covered with the center-type Haar-feature, furthermore, they contain the three discussed FIPs at the corners of the ROI (Fig. 5.6(b)).

LBP and HOG based classifiers can also be trained both to FIPs and whole code areas [13], and since they are also considered fast and accurate general purpose object detectors, evaluation of their performance on code localization is highly motivated. Moreover, 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.

The biggest advantage of full code area training compared to FIP-training is that it does not require any triplet formation rule or any other complex post-processing step.

5.3 Evaluation and results

A total number of six classifiers were trained, based on Haar-like features, LBP and HOG, both for FIPs and full code objects. For the FIPs, feature symmetry



(a) (b)

Figure 5.6: (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.

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. A 32×32 sample size was used, which is larger than the one of the reference method, since training to the whole code object requires finer sample resolution. The cascade topology was applied for the classifier instead of a tree, since it showed higher overall hit rate in [3], and left required hit rate and false positive rate at the default values for each stage, with a total number of 10 stages.

Classifiers were trained 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, ones were used that shift the FIP not more than one FIP width, which property is needed for the assumption of maximum expected distortion at the post-processing 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. 5.7).

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 pre-processing steps of the decoding task.

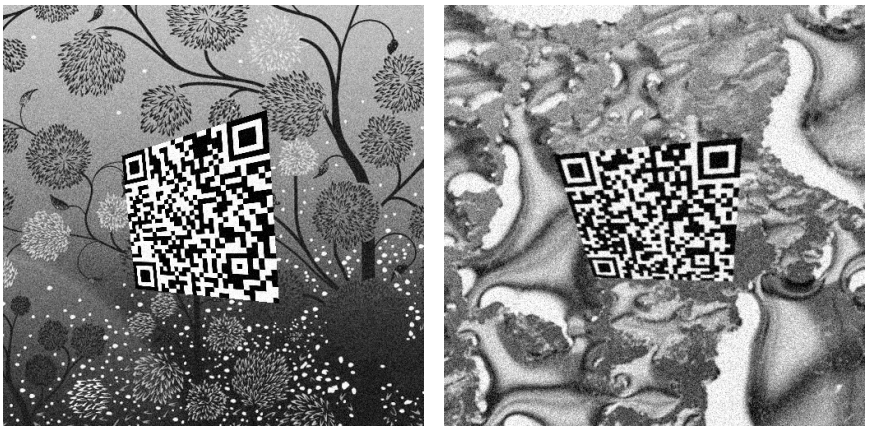
Training time for Haar features took cca. 15 hours on a Core 2 Duo 3.00 GHz CPU, while LBP training took about 1.5 hours, and HOG-training was the fastest, taking only about 30 minutes. There were only minor increases in training times of each category when training target was the full code object instead 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.

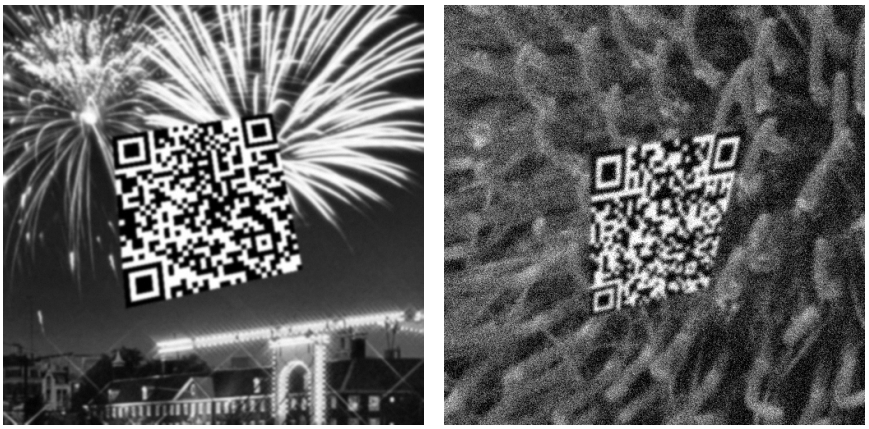
Table 5.1 shows performance measures of the examined cascade classifiers. HAAR-FIP, as stated by authors of [3], has a hit rate above 90 %, and represents a good solution for FIP-training. However, all FIP-based classifiers have poor precision compared to the ones trained for full code region, and they 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.

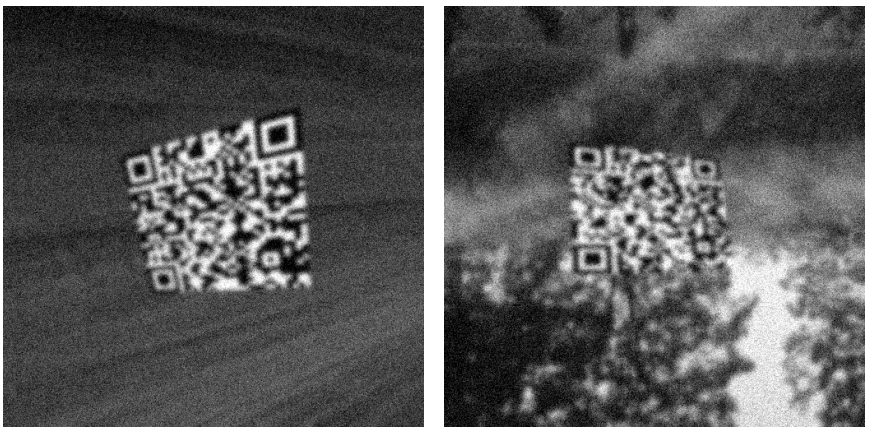
Table 5.2 shows results of the trained classifiers for the public database of Sörös et al. [67, 70]. HAAR-FULL, LBP-FULL and HOG-FULL are the same classifiers like in Table 5.1, they are trained only in our training database and were evaluated with no modifications. The last three classifiers, HAAR-SOROS, LBP-SOROS and



(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 5.7: Samples of the training database with different amount of smoothing and noise.

	Precision	Hit rate	F-measure
HAAR-FIP [3]	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 5.1: 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 5.2: Classifier performances for the database of Sörös et al. [67]. The ones ending with -FULL are the same classifiers trained on our synthetic database, while -SOROS classifiers are trained on their public database.

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.

Cascade classifiers were also trained 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, training two separate LBP-FULL classifiers with two training sample databases are recommended, with sample orientations around 0° and 45°, respectively.

5.4 Summary of the chapter

In this section, cascade classifiers were trained and proven to be applicable for the barcode localization task. A work on FIP-training using Haar-wavelets have been extended regarding both feature selection (LBP, HOG) and input data (full code training). Results are published in [7].

Chapter 6

Fuzzy inference systems

A simple approach to identify textures is using various stochastic measures [32], while Wang [33] introduced a more sophisticated texture filtering algorithm using Texture Spectrum as a general measure for texture properties, and Texture Units that express local intensity relations within an image cell. Lee et al. [42] even introduced the *Fuzzy Uncertainty Texture Spectrum* and Fuzzy Texture Units, a more generalized way of texture identification involving evolutionary algorithms and Fuzzy logic. However, the computation of these features would take a notable amount of time on embedded systems and make on-line processing very hard to implement. However, terms of Fuzzy theory can be applied in order to detect barcodes and Fuzzy Inference Systems (FIS) can perform evaluation rapidly, regarding the features they use.

6.1 FIS model parameters

The use of Fuzzy algorithms are already proven to be efficient for QR code localization [85]. A Fuzzy Inference System (FIS) is proposed in this chapter, based on the most simplistic, attentive features of a QR code, but this approach can be adapted to all 2D code types mentioned above [8].

6.1.1 Fuzzy features

I introduce an algorithm that can be efficient with respect to computation time and storage, and most of the computed features can be approximated using only a subset of pixels, that allows fine-tuning of the application to be faster or more accurate. These properties can make FIS-based localization a preferred choice over other fast algorithms.

After the code is located, there are reliable methods for correction of camera shaking and orientation [19], and correction of perspective distortion [52].

The proposed FIS consists of three input and one output variables. Membership function (MF) parameters are tuned each time to the end-user setup using statistics of a few input images. For the selection of properties, simple features that represent humanly observable properties are pursued. The three properties that can be summarized in the following statement: QR code parts consist of mostly black and white pixels of similar amounts, while having from moderate to high contrast and low saturation.

6.1.2 Membership functions

Since most QR code parts only have very dark and very bright intensities, the first variable is based on the absolute difference of pixels from the 50 % gray value. Intensity values referred in this thesis, are normalized to $[0, 1]$ from the 8-bit grayscale input images to fit the ranges of the membership functions. The first property is

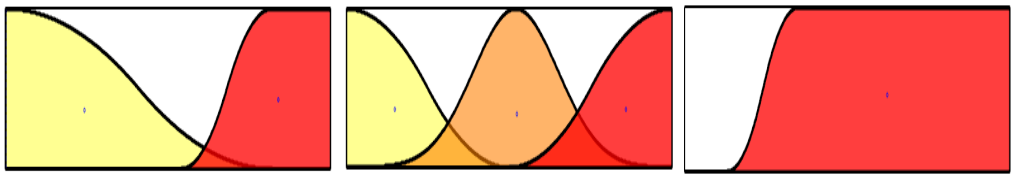
$$\text{graydist\_avg} = \frac{1}{n} \sum_{p \in \text{block}} |V(p) - 0.5| \tag{6.1}$$

where n denotes the number of sampled pixels from the block, and $V(p)$ is the V value of the pixel in HSV color space. Instead of using $(V(p) - 0.5)^2$ as the distance, $|V(p) - 0.5|$ is recommended because the membership function in the next step will give a soft boundary to this attribute anyway, and tolerance can be controlled by the steepness of the membership function.

From a couple of sample images, blocks being fully covered with QR code parts as positive samples, and blocks with 0 % coverage ratio as negatives, are extracted. After that, mean and standard deviation are computed for *graydist_ avg*. This was 0.41 ± 0.04 for positive samples and 0.31 ± 0.12 for negatives in our first test set. That would define two Gaussian membership functions, $perfect(m = 0.41, \sigma = 0.04)$ and $low(m = 0.31, \sigma = 0.12)$, however, using these would let very small tolerance and they would not cover the whole input range. To overcome this, Z-shaped and S-shaped membership functions are used instead of Gaussians (Fig. 6.1(a)).

A reasonable Z-term for *low* is $\text{ZMF}(0, 0.41)$, because the mean of *graydist_ avg* was at 0.41. For *perfect*, an S-term of $\text{SMF}(0.31, 0.41)$ is proposed, since negative sample mean was 0.31, which means, from that point, we have no information about the block content according to this property.

The endpoint of the S-term should be 0.41, since that was our measured mean



(a) (b) (c)

Figure 6.1: FIS input variables. (a): Mean brightness absolute difference from gray: *low* (yellow) and *perfect* (red); (b): Mean brightness: *low* (yellow), *perfect* (orange) and *high* (red); (c): Mean saturation: *high* (red). Parameters are indicated in the text.

value for the test images. This parameter would be 0.5 in the perfect case, and lower values reflect the amount of blurring present in blocks containing QR code parts.

The second input variable is *blockavg*, the mean intensity of the block. We obtained 0.52 ± 0.13 for positive, and 0.44 ± 0.31 for negative examples. Having this value around 0.5 for positive samples is expected because of the structure of the QR code. For negative samples, it is dependent on the content of the block. This property seems to have small classification power, however, having the value around 0.5 is a necessary condition for a positive sample. Three membership functions are proposed, one Gaussian for $perfect(m = 0.52, \sigma = 0.13)$ values, a $\mathrm{ZMF}(0, 0.5)$ and a $\mathrm{SMF}(0.5, 1)$ for *low* and *high* blocks, respectively (Fig. 6.1(b)). Both of the last two MFs express low certainty of presence of a QR code part within the block.

The third input parameter *saturation* excludes regions that have high saturation, since highly saturated areas are less likely to contain QR codes. The goal with saturation was to improve precision while keeping the hit rate. Mean saturation was 0.13 ± 0.05 for positive, and 0.39 ± 0.22 for negative samples, so a $\mathrm{ZMF}(0.13, 0.39)$ is proposed as *high*, shown in Fig. 6.1(c).

The output of the FIS is *codeness*, the certainty of QR code texture within a block, which can be expressed by two MFs, $\mathrm{ZMF}(0, 1 - x)$ and $\mathrm{SMF}(x, 1)$ with $x \in [0, 0.5]$, providing different level of smooth transitions. An intermediate value of $x = 0.33$ was used in the proposed model, thus producing $\mathrm{ZMF}(0, 0.67)$ and $\mathrm{SMF}(0.33, 1)$ for *low* and *high* MFs, respectively. As for the output, there is also an option of using the Takagi-Sugeno model instead of the Mamdani, however, manual fine-tuning the Mamdani model is easier in this case.

6.1.3 Rules

The rule set of the FIS contains the following rules:

R1 **if** *blockavg* **is** *perfect* **and** *graydist_avg* **is** *perfect* **then** *codeness* **is** *high*

R2 **if** *blockavg* **is** *low* **or** *blockavg* **is** *high* **then** *codeness* **is** *low*

R3 **if** *graydist_avg* **is** *low* **then** *codeness* **is** *low*

R4 **if** *saturation* **is** *high* **then** *codeness* **is** *low*

R1 is about positive response. Having the *blockavg* and *graydist_avg* properties in range are required conditions that have to be met simultaneously. The *blockavg* is needed so *high codeness* cannot be achieved with solid black or solid white blocks which both have *low saturation* and *high graydist_avg*. Following the same logic, it was assumed some contrast within blocks, expressed by *graydist_avg*, so solid gray blocks will not result in *high codeness*. R2 is to connect *low* and *high* MFs of *blockavg*, since both indicate QR code part is not likely in that block. R3 filters out blocks by *graydist_avg* in a similar manner like in the previous one. R4 is another exclusion filter, which is based on the mean of saturation, as discussed above.

6.1.4 Operators

For conjunction, *minimum* and a couple of product operators are available, like the *algebraic*, *bounded*, *drastic*, *Einstein* and *Hamacher* products. The product operators behave similarly, and show no significant difference in our case. *Minimum* operator is the simplest to compute. Both *blockavg* and *graydist_avg* has to be high for *high codeness*, and the decision based on the minimum of these variables suits our model. Products, like *algebraic product* can also be used, however, they lead to a stricter rule and steeper decision surface. Considering the precision from the results of each end-user setup, *algebraic product* can be preferred, since it might provide higher rate than *minimum* does, but *minimum* requires less computation, only a comparison instead of a multiplication.

For disjunction operator, the *maximum* is recommended with this rule set and MF layout, since that is the simplest operator to compute, and in this case, where no overlapping is present between the MFs participating in the conjunction (Fig. 6.1), there is no difference in the results of *maximum* and the various *x-sum* operators available. For the case of simplicity, *minimum* can be used for activation. The

usage of *algebraic product* would result in smoother transitions with the defuzzified output variable, however, it does not affect accuracy of the FIS significantly. However, using *algebraic product* over *minimum* might lead to performance drop respect to speed in time-critical applications. For accumulation operator, the *maximum* is recommended, since there are only two MFs of the output, and they are also symmetrically situated, thus there is no need to use complex operators at that step.

In the defuzzification step, *Centroid*, *Bisector*, or *Mean of maximum* can be used. According to the MF layout, there is no point in using *Smallest of maximum* or *Largest of maximum*. The *Centroid* or *Bisector* are recommended in this case, since they provides smooth transition of the output.

6.2 Evaluation and results

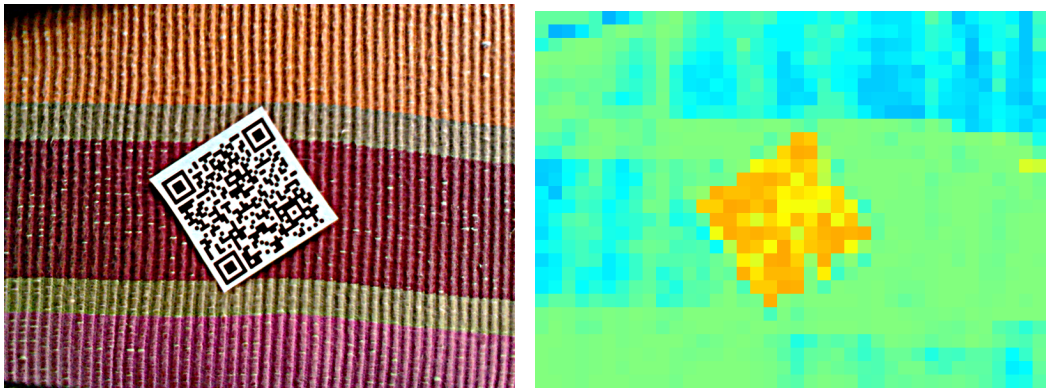
The proposed method has been evaluated on 98 arbitrarily acquired images using a 3.2 Mpx Huawei smartphone camera, without auto-focus capabilities and flash. Unit size of the QR codes present in those images were about 6–10 pixels, overall QR code size was cca. 200×200 pixels. Image size was 800×600 pixels. An example from this set is shown on Fig. 6.2. Even lighting is preferred, but not necessary for the captured images. Images with uneven lighting has to be pre-processed with local contrast stretching performed in each block.

Color images are also preferred for the saturation rule, that can be replaced by the rule based on histograms in case of grayscale input images. Geometrical distortions of the code leave the above features intact, as long as there are sufficient blocks to form a ROI in the feature matrix.

A Core 2 Duo 3.00 GHz CPU could process roughly 20 of these images each second, so real-time localization is possible. The FIS can be further optimized using ramp terms instead of ZMF and SMF. HSV channel images can be easily computed from RGB channels using $V_i = \max(R_i, G_i, B_i)$ and $S_i = (\max(R_i, G_i, B_i) - \min(R_i, G_i, B_i))/V_i$ for all $i \in I(x, y)$. Furthermore, using a lookup table instead of online calculations is also possible, since the table only would take one megabyte of data using precision of two decimals, which is sufficient for the task.

As the first test, various block sizes were evaluated to determine optimal block size ratio according to the expected QR code size, not involving histograms. Results show that optimal block size is ranging from about 20 to 35 percent of the QR code size (Fig. 6.3). Performance measures were based on the Jaccard measure.

Choosing too small block size leads to performance drop, since the attributes



(a) (b)

Figure 6.2: Printed QR code on tablecloth (a) and its FIS feature image (b).

Table 6.1: Evaluation of the FIS on different block offsets and 40 px block size.

Block offset	T_{opt}	F-measure	Precision	Hit rate	AUC
10 px	0.62	0.8124	0.8349	0.8766	0.8934
20 px	0.62	0.8124	0.8354	0.8773	0.8938
40 px	0.62	0.8584	0.8377	0.8802	0.8709

computed from the blocks become less reliable, while too large block sizes also decrease accuracy, since then only a smaller number of blocks are fully covered with a QR code part, and partially covered blocks are also harder to classify. Instead of a fully universal, multi-scale solution [49], specific resolutions and block sizes lead to more accurate implementations that can be important on embedded systems.

The effect of the block overlap to performance was also evaluated and is shown in Table 6.1. Block size was set to 40 px and each block was offset by 10, 20 and 40 px (meaning no overlap), respectively. Results show that evaluation with overlapping blocks did not increase performance.

Performance of the FIS has also been evaluated on code types other than QR codes. We assembled four test sets containing Aztec codes and Data matrix codes

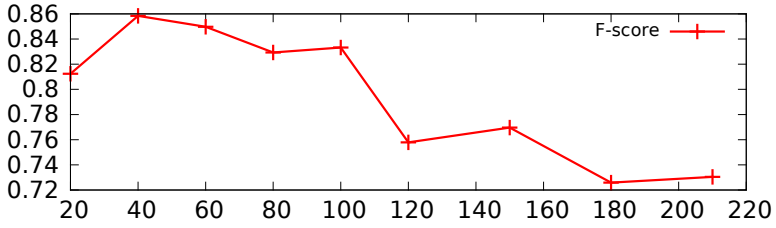


Figure 6.3: Performance of the FIS with respect to block size.

Table 6.2: FIS performance on different visual code types. 1D-S denote for stacked barcodes.

Type	Dim.	Block size	Precision	Hit rate	F-measure
QR	2D	50 px	0.9224	0.9353	0.9288
Aztec	2D	50 px	0.8738	0.9639	0.9167
Data matrix	2D	50 px	0.9214	0.9399	0.9305
Codablock	1D-S	20 px	0.7136	0.7287	0.7210
PDF417	1D-S	20 px	0.7277	0.7007	0.7140

as two-dimensional, and Codablock and PDF417 codes as stacked one-dimensional types. Stacked 1D codes, like real 2D ones, embed information along both axes. These synthetic examples are built with computer-generated codes containing random letters and numerals of the alphabet. The code was placed on a negative image, with random rotation. Gaussian smoothing and noise have been gradually added to the images. The σ for the Gaussian kernel was varied in the range [0,3]. A noise image (I_n) was generated with intensities ranging from [-127, 127] following normal distribution, and 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.

Results show that real 2D codes behave similarly to QR codes, despite their structural differences (Table 6.2). One-dimensional stacked codes had smaller height, therefore the block size has been set to smaller, though the localization performance was inferior to that for real 2D codes. This is probably due to the fact that in real 2D codes row and column patterns are similar while in stacked 1D codes they are quite different. The input variables used for the FIS are basically direction-invariant and thus suit better for 2D codes.

To compare efficiency of the proposed method to other implementations from the state of the art, it was evaluated on two public databases, from Sörös et al. [67], and Dubská et al. [23], respectively.

Sörös et al. made their set using 200 blurry images acquired by iPhone5, without auto-focus. The evaluation of the FIS on this set was performed with minor modifications of the original input terms based on sample images of the set. The *graydist_avg* attribute had its *perfect* SMF term adjusted to $\mathrm{SMF}(0.25, 0.5)$, since images of this set had poor contrast due to the heavy blur present. Mean intensity of the blocks were also higher, so the Gaussian term representing the *perfect* membership function has been modified to $G(m = 0.61, \sigma = 0.09)$. SMF term regarding saturation could be set to $\mathrm{SMF}(0.075, 0.19)$, which led to a stricter saturation rule

Table 6.3: Results of the proposed method on the Sörös et al. and Dubská et al. data sets.

Data set	Precision	Hit rate	F-measure
Sörös et al. (Original FIS)	0.5938	0.5224	0.5558
Sörös et al. (Median filtered)	0.5890	0.6018	0.5953
Dubská et al. Set-1	0.6165	0.5036	0.5544
Dubská et al. Set-2	0.9288	0.9513	0.9399

than the one of our original test set. Results in Table 6.3 show performance of the FIS on this test set for the original algorithm, and one with median filter as post-processing. Fig. 6.4 shows an example of this data set with the corresponding feature images.

The second public database by Dubská et al. contained two similar sets of QR code images, surrounded with text in a scene having low saturation in general. The first set has 410 high-resolution (2560×1440 px) images with uneven lighting conditions, high grades of distortion and minor blur (Fig. 6.5(a)). The second test set has 400 low-resolution (604×402 px) images with smaller grades of distortion and more even illumination, but having less light in general, thus producing darker images (Fig. 6.5(c)).

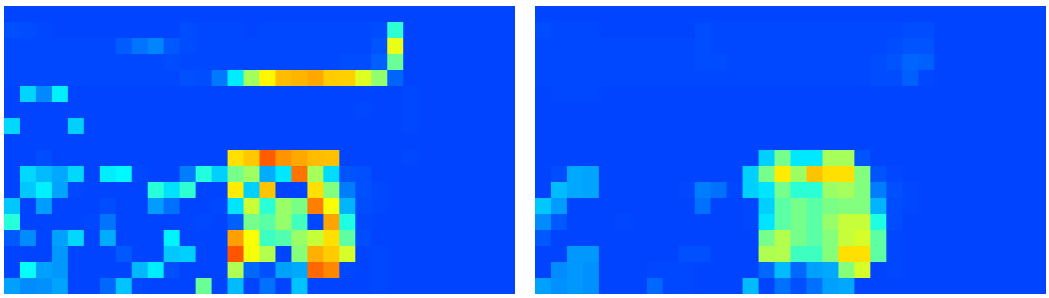
For the first set of this database, FIS had to be set for larger tolerances for the *perfect* term of *blockavg*, and *graydist_avg* was also set to lower acceptance value, defined by SMF$(0.25, 0.3)$. However, images of the first set have shown so high variability for the mean intensities within blocks, contrast and QR code size that the designed FIS could not be generalized enough to classify all samples well. We can overcome this issue using adaptive thresholding or local contrast stretching, at the cost of more computation time.

On the second data set, with a chosen block size 50 px, FIS terms of positive response could be tuned more easily, therefore the proposed method performed better with respect to both precision and hit rate (Table 6.3).

6.2.1 Using partial block information

As in the case of histograms, the number of read pixels can be limited to speed up FIS processing. Fig. 6.6 shows results of the sparse data evaluation. The x axis represents the sampling factor, which means that only every n-th pixel is read from both the rows and columns, so the amount of pixels used to calculate the FIS input variables, is reduced by a factor of n^2. This partial block information does not

(a)



(b) (c)

Figure 6.4: Output stabilization of a sample image from the Sörös et al. set. (a) original image, (b) feature image, (c) median filtered feature matrix.

introduce more false positives, it only affects the hit rate by rendering the only rule of positive response unreliable in the FIS.

Results also show that the chosen sampling interferes with the unit size of the QR code. Hit rate temporally rises while reading only every 10th, 12th and 15th pixel, since it gives a more reliable block sampling for attribute computation, which is caused by the cca. 6 px unit size of the used QR codes in those images. Choosing the sampling factor $k \cdot \text{unitsize}/2, (k \in \mathbb{Z})$ is more likely to sample most QR code units from the same position, like close to center of unit, or close to their perimeter. However, assumptions can not be made on expected QR unit size on arbitrarily acquired images, therefore using large sampling factor is considered unreliable in general.

Furthermore, chosen block size gives an upper limit to sampling, since calculation of the FIS input attributes are based on statistics and thus, require tens of pixels for each block.

In order to stabilize the FIS output by region compactness, adjacent blocks have to be taken into account. To avoid large increase of computation time, evaluation of this condition is recommended to be performed outside the FIS, in the feature matrix. Small "holes" of the matrix, values surrounded by blocks of high values, are

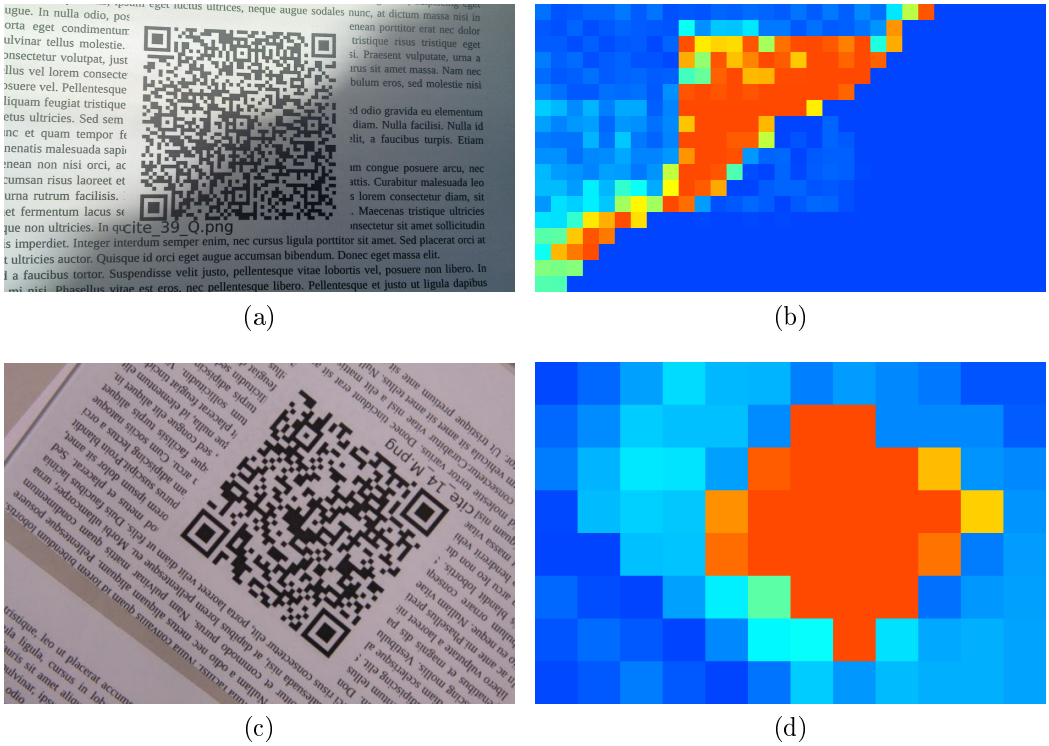


Figure 6.5: Examples of the Dubská et al. Set-1 (a) and Set-2 (c), and their feature images (b) and (d), respectively. In both cases the block size was 60 px, but the size of the first image is much higher than that of the second.

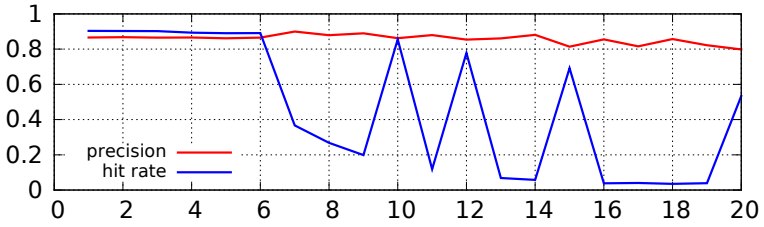


Figure 6.6: Performance of the FIS on sparse data. Values of the x axis mean that only every n-th pixel is read with respect to rows and columns.

likely to be false negatives. Similarly, "lonely" blocks of high value can be safely zeroed out, since they probably do not participate in any QR code candidate. Morphological filtering, or as a simpler operation, median filtering are suitable for this task (Fig. 6.4). In most cases, the latter seems sufficient according to experimental results, however, using mathematical morphology at this step is also acceptable, as the size of the feature matrix is only a small fraction of that of the original input image.

6.2.2 Extension of the attribute set

At some setups, color information might be unavailable or insignificant. By omitting the saturation rule, further improvements on speed can be achieved, however, F-measure dropped from 0.8084 to 0.7408, which is caused by the increased false positives that the saturation rule could filter. Median filtering of the feature image is an option in this case, like at partial data processing, but not sufficient for efficient localization in most cases. To overcome this issue, a new attribute can be introduced to replace the saturation rule. By reading pixels from each block along a circle having radius of half the block size, a one-dimensional vector can be obtained.

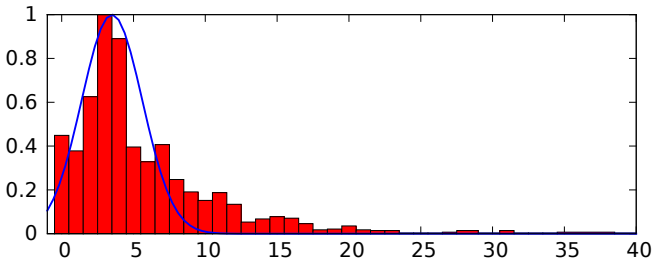
After the binarization of the block, specific run lengths of black and white pixels are observable if a QR code part is present, and more random ones for blocks with no code (Fig. 6.7). Blocks containing QR code parts have slightly larger mean values than ones with neutral content, which ensues from the orderly structure of QR codes. The attribute formed using this circular pattern is not sensitive to the orientation of the QR code, but it fails in cases of high perspective distortion, or when the localization pattern of the QR code is present within the block.

Two Gaussian distributions can be given as an approximation for the positive and negative samples, showing this feature has fairly low classification power, however, it can stabilize accuracy when the saturation rule is not applicable. The FIS input variable designed on this feature is based on the Earth Mover's Distance (EMD) [59] of the actual sample and the empirically measured positive one. The smaller is the distance, the stronger is the feature, and distance values can be normalized by the EMD of the cumulative distributions of positive and negative samples (Fig. 6.8), since this feature shows no additional information regarding QR code presence when EMD is larger than that.

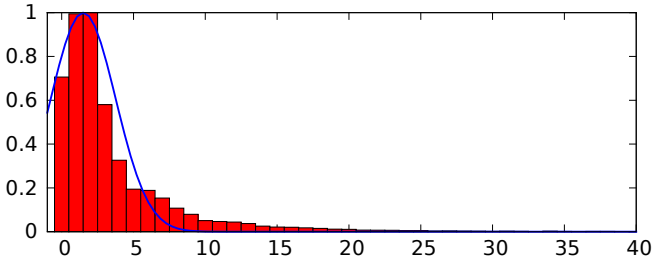
Hence, using this feature is recommended only as an exclusion filter, replacing the saturation rule when necessary. A simple S-term can be used, (Fig. 6.9) as it will filter out blocks having large EMD values to the ideal case. EMD values can be limited to the distance of mean positive and negative cumulative distributions, since the feature represents no additional information regarding block content from that point.

Although the first FIS rule would fire in most cases when this feature indicated positive response, thus making addition of another positive rule unnecessary. Low EMD suggest similarity to a QR code part within the block. The final rule follows as "if RL-measure is high, then codeness is low".

In some cases, this new attribute can be used together with the saturation rule



(a) Positive blocks



(b) Negative blocks

Figure 6.7: Distributions of runlengths along the circlular pattern, using cca. 1000 positive and negative samples.

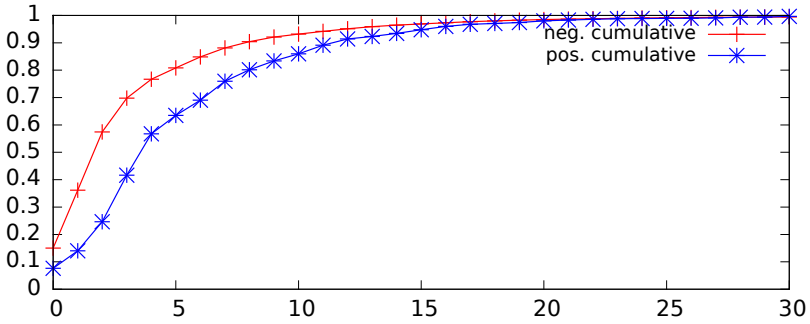


Figure 6.8: Mean cumulative distributions of positive and negative samples.

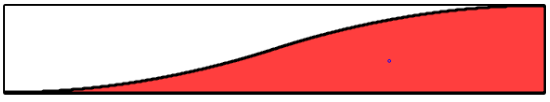


Figure 6.9: Runlength S-term, $S(0, D(\text{pos}, \text{neg}))$.

if the application needs more exclusion filters, however, it would make lookup tables one scale larger, to at least 100 Mbytes, which is not suitable for some embedded systems.

6.2.3 Remarks on hierarchical processing and hybrid solutions

All three attributes from the basic concept of the proposed FIS make hierarchical processing available. Using integral images for S and V channels of the HSV space, and a third one constructed from the channel of $D, D_{x,y} = |V_{x,y} - 0.5|, V_{x,y} \in [0, 1]$. Using the integrals of S, V and D channels, features for blocks of any size can be computed in constant time, and features for n scales to only $\mathcal{O}(n)$. Multi-scale processing is therefore available and desired, though only scaling of the block size used for the FIS is recommended, but not for input construction, since integral images provide a fast and reliable way of using all the pixel data available. Using sparse patterns for FIS input should only be preferred when processing speed is top priority, or input image resolution is very high.

This implementation would also let evaluation with dense overlapping of the processed blocks regarding computation of the FIS inputs, and the FIS itself can be replaced by lookup table to avoid time-consuming operations, thus raising the whole system to the convolutional level. Moreover, it does not significantly increase efficiency of the method, and instead of overlapping larger blocks, evaluation of smaller, distinct blocks is more desirable due to the last step of the proposed method, the connected component lookup.

It is important to note here, too, that feature images of different approaches can be combined, as it is seen in Section 3.6. Instead of the weighted sum of these images, Fuzzy operators give more sophisticated options for aggregation. Individual feature images can be considered as membership functions, and rules can be defined to increase overall accuracy. However, it is not straightforward to define the weights for the features, since each feature set might have correlations between features. FIS is not capable of learning by itself, but Fuzzy approaches are often combined with learning algorithms, like neural networks and evolutionary algorithms.

In case of Neuro-Fuzzy implementations, a neural network learns the importance level of each fuzzy rule and optimizes the weight vector, which is usually performed by back-propagation. Evolutionary algorithms are highly flexible. Each member of the population can represent a single fuzzy rule or a rule set. In the first case, the fitness reflects to the importance or classification power of that rule, while in the

other case, fitness is the efficiency of a rule set. Rule sets can be selected randomly and evolved using the classical concepts of reproduction and mutation, or they can be pre-processed using data mining approaches. One of the most common examples is to select features and derive new ones using algebraic operators (+, -, *, /), and evaluate the importance of these new features with respect to their classification power.

6.3 Summary of the chapter

In this section, Fuzzy Inference Systems were introduced for the localization task. Fuzzy Inference Systems are capable of performing both the localization task itself [8], and aggregation of feature images individually produced by different algorithms. Moreover, Fuzzy Inference Systems are fast and can be implemented easily.

Summary

The use of visual codes became wide-spread in our lives, both in industrial applications and private projects. Compared to other technologies like RFID, they are easier to use and less expensive. With the improvement in technology regarding both hardware and software systems, there has been a rise in the interest towards the automatic recognition of not only the well-positioned and oriented codes, but also towards all codes within the sensor area, in various sizes and orientations. This task is now available to accomplish in real time, thanks to the evolution of computer hardware. For more simple embedded systems, algorithms that require small computational power are desired.

For setups that allow more complex hardware, algorithms using machine learning techniques and various classification methods are now available. However, all these algorithms are balancing between processing speed and accuracy. Each end-user application can define which one of this two is more important. A barcode reader smartphone application can be more inaccurate, and may require taking more shots or better positioning, while an industrial reader program might require maximum accuracy and autonomous work.

The decoding of the embedded data happens in two steps. The first step is the localization of the code object within the observed area, while the second one is the decoding of the visual information from the code candidates. The latter step can be considered straightforward, if the code candidates are localized and pre-processed accordingly. The aim of this thesis is to give improvements for existing localization methods, and give new approaches is to consider new machine learning techniques and opportunities provided by the improved hardware.

After a brief introduction, fast algorithms are presented in Chapter 2 based on simple features using global information, with proposed improvements. In the next chapter (Chapter 3), I introduced new algorithms and modified the scan-line approach to work with image tessellation. In Chapter 4, neural networks are presented for the localization task, examine their efficiency and fine-tune their param-

eters. The behavior and capabilities of the conventional and deep rectifier networks are evaluated. The emphasis was put on the examination of neural networks that can be trained directly in the DCT domain. This greatly speeds up the classification process for JPEG images. Chapter 5 examines efficiency of cascade classifiers which solves the classification task using a set of weak features. Chapter 6 involves Fuzzy Inference Systems, which describes the problem and makes decisions using tools similar to ones of the probability theory.

The earliest algorithms are based on global information and simplistic features. Such algorithms required small computational power, but manual assistance was often needed during their execution. A typical example is the software of manual barcode readers at Point of Sale terminals. These algorithms had strict conditions to the code size and orientation, since they only worked accurately in small ranges of the parameter values. However, they gave the core idea for later algorithms that imitate and improve the operation of laser scanners. This line-based processing can be applied for the whole image. The one-dimensional intensity profiles obtained this way, mark the sections that are likely to contain visual code parts. The number of these sections and their position to each other defines regions of interest within the image that can be cropped using rotated bounding boxes, and forwarded to the decoder that retrieves the embedded data. For finding line segments, Hough transformation and mathematical morphology are more efficient ways than the scan-line approach. Using mathematical morphology, one can enhance areas that might contain code segments, and suppress ones that might not. With a couple of post-processing steps ROIs can be given in those feature images.

A group of algorithms that use tessellation of the image, can be separated. The input image is divided into square blocks, and each block is assigned a measure using pixel information of the block. This measure reflects the likelihood of visual code presence within that block. The matrix formed by these values can also be considered as a feature image, where one can find compact areas of high values. Regions of interests defined by these components are forwarded to the decoder. Various algorithms are based on this concept. The most simplistic one is local intensity clustering, which requires reading all pixels of a cell. A more efficient method is circular scanning, which is analogous to the scan-line method, but invariant to code orientation, and requires only a fraction of the pixels to be read for the feature. A similar method to line scanning is the runlength measuring, that works for both 1D and 2D cases. Furthermore, there are some other approaches that are also considered simplistic, like feature extraction from cell histograms, and distance transformation performed cell-wise.

Another approach to the localization methods involves machine learning, specifically neural networks. Chapter 4 discusses both the conventional and deep neural networks. Deep rectifier networks, which have the same expressive power as the normal ones at fixed number of neurons, have better performance measures in most applications. Neural networks require learning, which is usually performed by the back-propagation algorithm. However, back-propagation is less effective in the case of deep networks, and require modifications. The efficiency of the networks is evaluated using various input types. Vectors are extracted from image cell information. Deep rectifier networks were also able to learn efficiently in the DCT domain, which enables training using JPEG images, without the necessity to perform the most complex step of the JPEG decoding process, the inverse DCT. The process can be further speeded up if one has information about the compression level of the images at the end-user application, therefore the de-quantization step can also be omitted. Furthermore, the regression learning and domain adaptation capabilities of the deep networks were also evaluated.

The last group consists of classifiers that use more more complex features than the first group, and yet they are fast and easy to implement. One of the popular, general purpose classifiers is the cascade classifier that uses weak features, presented with improvements in Chapter 5. It can also be used with Haar-like features, Local Binary Patterns or Histogram of Oriented Gradients. Literature already discusses localization with Haar-like features trained on the finder patterns of QR codes. In this thesis, a more efficient method is introduced, that has a learning phase on the full code objects. It also makes post-processing unnecessary, since it does not require triplet formation of the finder pattern candidates in order to give code object candidates.

Besides cascade classifiers, using Fuzzy Inference Systems (Chapter 6) is also an alternative. Some simple features are proposed, that can be defined with membership functions, and I also gave more features for cases where simple features do not provide the required accuracy. Fuzzy Inference Systems that work with few features can be implemented to react in constant time, since their knowledge can be transformed into lookup tables. Combination of Fuzzy systems with existing approaches and Fuzzy processing of the feature images are also discussed.

These approaches, the algorithms based on easily computeable global or local features (Chapter 2 and 3, respectively), the neural networks (Chapter 4), the boosted classifiers (Chapter 5) and the Fuzzy Inference Systems (Chapter 6) cover the localization topic with respect to the requirements of accuracy and speed. They give solutions and applications that have various hardware systems and expectations.

Thesis summary

1. I introduced 3 new algorithms for barcode localization in image using global information. These, respectively, are based on scan-line analysis with new features, Hough transformation, and mathematical morphology operations. The latter two outperform state-of-the-art algorithms in terms of accuracy and recall, while the scan-line approach is very fast and is also reasonably accurate.

2. I proposed new algorithms for the localization of visual codes using the idea of image tessellation. The proposed algorithms have low computational and storage requirements, and can be easily adapted for parallel computation.

   2.1. I proposed cell histograms, distance transformation and a modified scan-line approach for local feature formation. I have also shown that an ensemble of simple detectors can have higher precision or recall than state-of-the-art algorithms, depending on the type of aggregation and involved features.

   2.2. I implemented a rotation-invariant feature adapted from the scan-line approach. The new feature uses local intensity profiles read along a circular pattern, and takes advantage of symmetries and neighboring cell information.

3. I introduced and evaluated neural networks for visual code localization in images.

   3.1. Through the experiments I have shown that the use of Deep Rectifier Networks is a viable option for barcode localization, both in image space and in the frequency domain, even with binary images.

   3.2. I have also shown that Deep Rectifier Networks can be trained on JPEG DCT vectors, that eliminates the need of the inverse DCT step of the JPEG decoding process.

4. I evaluated the performance of cascade classifiers and their usability for barcode localization, and proposed new extensions.

   4.1. I proposed 2 new features for the training, namely, Local Binary Patterns (LBP) and Histograms of Oriented Gradients (HOG). They are proven to be more accurate compared to Haar-wavelets, which is presented in cascade classifiers from the state-of-the-art.

4.2. I proposed learning on the full code object instead of learning on the finder patterns exclusively, which greatly simplified post-processing.

5. I introduced Fuzzy Inference Systems for barcode localization, that provides fast execution and a flexible model construction.

Összefoglalás

A vizuális kódok használata igen elterjedt a mindennapokban, mind ipari és magáncélú alkalmazási területeken. Használatuk olcsó és könnyű más technológiákhoz, például az RFID-hez képest, ugyanakkor felismerésük kevésbé lehet pontos. A technológia fejlődésével igény támadt arra, hogy nem csak a céleszköz számára megfelelően pozícionált és orientált 1D és 2D kódokat ismerjük fel automatikusan, hanem a megfigyelt területen belül tetszőleges méretben és elhelyezkedésben is. A számítógépes hardver fejlődése lehetővé tette a feladat megvalósítását valós időben. Egyszerűbb beágyazott rendszereken gyorsan végrehajtható, kevés számolást igénylő algoritmusokra van szükség, míg egy bonyolultabb számítógép gépi tanulási módszerekkel és komplexebb algoritmusokkal is végezheti a feldolgozást. A jelenlegi algoritmusok mindegyike egy kompromisszum a sebesség és a pontosság tekintetében. Azt, hogy ezek közül melyik a fontosabb kritérium, a végfelhasználói alkalmazás határozza meg. Egy okostelefonra írt vonalkód-leolvasó alkalmazás lehet pontatlan, igényelheti több kép elkészítését, vagy jobb pozícionálást, míg egy ipari projekt megkövetelheti a maximális pontosságot és a felhasználói beavatkozás nélküli működést.

A kódokba ágyazott adat visszanyerése két lépésben történik. Első lépés a megfigyelési térben a kód-objektum megkeresése, majd a megtalált objektumból az adatok dekódolása. Az utóbbi lépés egy megfelelően előfeldolgozott kód alapján egyszerűnek tekinthető. A dolgozat célja meglévő lokalizálási eljárások továbbfejlesztése, illetve új módszerek kidolgozása, figyelembe véve a hardver fejlődése adta lehetőségeket.

A dolgozat fejezetei a következők. Egy rövid ismertető után a 2. fejezetben egyszerű, globális képi jellemzőkön alapuló algoritmusok bemutatása történik meg, fejlesztési lehetőségeket és új algoritmusokat is adva.

Az egyszerű jellemzőkön alapuló algoritmusok a legkorábbiak. Ezek még kevés számítási igénnyel rendelkeztek, és legtöbbször emberi beavatkozásra volt szükség a működésükhöz. Tipikus példája az áruházakban megtalálható kézi vagy tükrös rendszerű vonalkód-leolvasó készülékek szoftvere. Ezek az algoritmusok megkötéseket

tartalmaztak a kód méretét és pozícióját tekintve, és adott tartományokon kívül nem működtek hatékonyan, mégis ezek a módszerek adták az alapötletét néhány későbbi algoritmusnak, melyek a vonalszkennerek módszerét fejlesztették tovább. Az egész képtérre kiterjeszthető ez a soronkénti feldolgozás. Az így nyert egydimenziós intenzitás-profilok alapján megjelölhetőek olyan szakaszok, melyek nagy valószítűséggel vizuális kódot tartalmaznak. Ezen szakaszok mennyisége és egymáshoz viszonyított elhelyezkedése alapján megadhatjuk azokat a területek a képen, melyeket egy befoglaló téglalappal kivágva és opcionális korrekciós lépéseket elvégezve továbbítunk a dekódernek, mely az adatok visszanyerését végzi. A vonalszkenneléstől hatékonyabb módszer szakaszok keresésére a Hough-transzformáció, ami szintén alapját adhatja a lokalizációnak. A kép szekvenciális, soronkénti feldolgozása helyett morfológiai műveletekkel is állíthatunk elő jellemzőképeket, melyek kiemelik azokat a területeket, melyek kódot tartalmaznak, és elnyomják azokat, amelyek nem. Ezeket a képeket utófeldolgozva szintén megtalálhatjuk a fontos régiókat a képen.

A 3. fejezetben kitüntetett figyelmet fordítok a mozaikfelbontáson alapuló, lokálisan számolható képi jellemzőkre, mivel bizonyos esetekben nincs lehetőségünk a teljes képet tárolni, a szenzorból érkező új információ felülírhatja a régebbit. A képet egybevágó, négyzet alakú cellákra osztjuk fel, és minden cellához egy mérőszámot rendelünk a cellában fellelhető pixelek alapján. Ez a mérőszám arra utal, mennyire valószínűsíthető egy vizuális kód részlete az adott cellában. A cellák mátrixa lényegében egy jellemzőképet ad, melyen összefüggő, magas értékkel rendelkező foltokat keresünk. Ezek a komponensek kerülnek tovább a dekóderhez. Erre az alapötletre számos algoritmus épül. A legegyszerűbb a cellán belüli intenzitás-alapú komponenskeresés. Ez az összes pixel beolvasását igényli. Ennél hatékonyabb módszernek minősül a vonalszkennelés alapötletével analóg, kör menti szkennelés, mely invariáns a kód orientációjára, és a pixelek számának töredéke elég a jellemző kinyeréséhez. A dolgozatban kiterjesztettem a klasszikus vonalszkennelési elvet csempékre. A hagyományos vonalszkennelés lokális változata a lokális futamhossz mérés, 1D és 2D esetben, és az ebből származtatott algoritmusok. Ezen kívül egyszerű módszernek minősül még a cellahisztogramokon alapuló jellemző-kinyerés, illetve a cellákban külön-külön végzett távolság-transzformációra épülő algoritmus.

A 4. fejezetben a neurális hálók használatát mutatom be a loklaizálási feladatra, elemzem azok hatékonyságát, megvizsgálom a lehetséges paramétereket. Összevetem a hagyományos és a mély neurális hálók viselkedését és képességeit. Külön figyelmet érdemel a neurális hálók DCT térben történő tanítása és az osztályozási feladat végrehajtása közvetlenül a frekvenciatérben.

A mély hálók, bár kifejező erejük azonos neuronszám mellett egyezik a konven-

cionális hálókéval, mégis a legtöbb alkalmazási területen jobb mérőszámokkal rendelkeznek. A neurális hálók tanítást igényelnek, mely a hagyományos esetben leggyakrabban backpropagation segítségével történik, mély hálók esetében ez kevésbé hatékony, és módosításokat igényel. Megnézem továbbá a különböző típusú input vektorok tanulhatóságát. A vektorokat továbbra is a képet felosztva kapjuk lokális információból. Ezek a mély hálók a DCT térben is hatékonyan taníthatók, ez által lehetőség nyílik JPEG képeken végezni a tanítást, anélkül, hogy a képek dekódolásának legnehezebb lépését, a DCT inverzét ki kellene számolnunk. A folyamat tovább gyorsítható, ha ismerjük a végfelhasználói alkalmazásban használt képek tömörítését, így tanításkor és kiértékeléskor már a de-kvantálási lépés is elhagyható. Megvizsgáltam továbbá a részben fedett blokkok kérdését, a mély hálók regressziótanulási és domain adaptációs képességét is.

A 5. fejezet kaszkádolt osztályozók hatékonyságát vizsgálja, amely gyenge jellemzők sokaságának segítségével valósítja meg az osztályozási feladatot. Ilyen osztályozók tanítását már tárgyalta a szakirodalom, fejlesztési lehetőség viszont a Haar-waveleteken alapuló jellemzők cseréje LBP illetve HOG alapúra. Ezen kívül említést érdemel a teljes kódrégió tanítása a lokátor minták tanításával szemben. Ez a javasolt módosítás szükségtelenné teszi a számításigényes utófeldolgozást és stabilabb osztályozást tesz lehetővé.

Hatékony módszer még a Fuzzy rendszerek használata (6. fejezet), amely a valószínűségszámításhoz hasonló eszközökkel írja le a jellemzőket és hoz döntést. A vizuális kódok felismerésekor gyakran nem tudunk, vagy nem szándékozunk konstansokat és küszöbértékeket használni, helyette relaxáltabban definiáljuk a feladatot és a megoldást Fuzzy elvekkel keressük. A fejezet igazolja, hogy ez működőképes elv a feladat megoldására. Részletesen tárgyalom az operátorok és a szabályok használatát, és javaslatot teszek további jellemzők bevonására azon esetekre, amikor a legegyszerűbb jellemzők használata nem biztosít kellő pontosságot. A Fuzzy rendszerek kevés jellemző esetén konstans időben osztályoznak, mert kevés memóriával lookup táblává alakítható a tudásuk. Teszek javaslatot a Fuzzy rendszerek meglévő algoritmusokkal történő kombinációjára, a jellemzőképek Fuzzy feldolgozására is.

A három fő téma, a gyors, egyszerű jellemzőt használó algoritmusok, a neurális hálók és az egyéb osztályozók használata a lokalizálás problémáját mind a sebesség, mind a pontosság nézőpontjából vizsgálja, és hatékony megoldási javaslatokat ad a változatos hardveres felszereltségű és különféle elvárásokat támasztó felhasználási területeken.

Ezek a megközelítések, a könnyen számolható globális és lokális jellemzők (2. és 3. fejezetek), a neurális hálók használata (4. fejezet), a boosting technikával javított

kaszkád osztályozók használata (5. fejezet) és a Fuzzy rendszerek (6. fejezet) lefedik a lokalizációs problémát mind pontosság és sebesség tekintetében, hatékony megoldási javaslatokat adnak a változatos hardveres felszereltségű és különféle elvárásokat támasztó felhasználási területeken.

A disszertáció eredményeinek összefoglalása

1. Bevezettem 3 új, globális információt használó algoritmust a vizuális kódok lokalizációjára képeken. Ezek rendre a klasszikus vonalszkennelésen, Hough transzformáción, valamint matematikai morfológiai operátorok használatán alapulnak. Az utóbbi két algoritmus képes felülmúlni a szakirodalom algoritmusait pontosság és találati arány tekintetében, míg a vonalszkennelésen alapuló algoritmus gyors és szintén elfogadható pontossággal rendelkezik.

2. Új algoritmusokat javasoltam a vizuális kódok helyének meghatározására a kép mozaikfelbontásának ötletét felhasználva. Ezeket az algoritmusokat alacsony számítási és tárigény jellemzi, valamint könnyen párhuzamosíthatók.

   2.1. Javasoltam a cellahisztogram, a távolságtérkép és a módosított vonalszkennelés megközelítését helyi jellemző-kinyerésre. Megmutattam továbbá, hogy az egyszerű jellemzők kombinációja képes felülmúlni pontosságban vagy találati arányban az eddig ismert algoritmusokat, attól függően, hogy milyen típusú aggregációt és jellemzőket választunk.

   2.2. Implementáltam egy forgás-invariáns jellemzőt, mely a klasszikus vonalszkennelésből származik. Az új jellemző egy kör mentén kinyert lokális intenzitásprofilokra épül, továbbá kihasználja a szimmetriákat és a szomszédos cellainformációkat.

3. Bevezettem és elemeztem a neurális hálókat a vizuális kódók lokalizációjára.

   3.1. Vizsgálataim igazolták, hogy a mély egyenirányított hálók hatékonynak bizonyultak vizuális kódok lokalizációjára, képtérben és frekvenciatérben egyaránt, bináris képeken is.

   3.2. Megmutattam továbbá, hogy a mély egyenirányított hálók közvetlenül JPEG DCT vektorokon is taníthatók, ami szükségtelenné teszi a JPEG dekódolás legköltségesebb műveletét, az inverz DCT-t.

4. Kiértékeltem a kaszkádolt, gyenge jellemzőkön alapuló osztályozók használhatóságát a vonalkód lokalizációs feladatra, és fejlesztéseket javasoltam.

4.1. Két új jellemzőt javasoltam az osztályozók tanításához, az LBP (Local Binary Patterns) és HOG (Histograms of Oriented Gradients) jellemzőket. Ezek bizonyítottan nagyobb pontosságot biztosítanak a szakirodalom tárgyalt, Haar-waveleteken alapuló jellemzőkhöz képest.

4.2. Javasoltam a teljes kódobjektumon végzett tanítást a kizárólag lokátormintákon végzett tanítással szemben, ami nagyban egyszerűsíti az utófeldolgozási lépést.

5. Bevezettem a Fuzzy következtetési rendszerek használatát a vonalkód lokalizációs feladatra, ami gyors végrehajtást és rugalmas modellalkotást tesz lehetővé.

Acknowledgement

The research was partially supported by the OTKA K112998 grant of the National Scientific Research Fund. It was also supported by the European Union and the State of Hungary co-financed by the European Social Fund under the grant agreement TÁMOP-4.2.2.B-15/1/KONV-2015-0006.

I would like to express my gratitude to my advisor, László G. Nyúl, for his active engagement throughout my studies, and providing me with useful comments, and remarks regarding this thesis. Furthermore, I would like to thank my excellent colleagues, Tamás Grósz, for the prosperous co-operation, and Melinda Katona for her passionate attitude and ideas concerning the topic itself.

I would like to thank my mother, without her continuous support and constant encouragement I would have never been able to complete my thesis or achieve my goals. I will be grateful forever for all your love.

Lastly, I would like to thank Suzana Laličić for her devoted assistance proof-reading the text of this thesis.

Publications of the author

Publications accepted by the PhD School in Computer Science

Journal publications

Péter Bodnár and László G. Nyúl. Improved QR code localization using boosted cascade of weak classifiers. *Acta Cybernetica*, 22:21–33, 2015

Péter Bodnár and László G. Nyúl. Barcode detection using local analysis, mathematical morphology, and clustering. *Acta Cybernetica*, 21:21–35, 2013

Conference publications

Péter Bodnár and László G Nyúl. Localization of visual codes using fuzzy inference system. In *VISAPP 2015 Proceedings of the 10th International Conference on Computer Vision Theory and Applications*, pages 345–352. SciTePress, 2015

Péter Bodnár and László G Nyúl. QR code localization using boosted cascade of weak classifiers. In *Image Analysis and Recognition*, pages 338–345. Springer International Publishing, 2014

Tamás Grósz, Péter Bodnár, László Tóth, and László G Nyúl. QR code localization using deep neural networks. In *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on*, pages 1–6. IEEE, 2014

Péter Bodnár, Tamás Grósz, László Tóth, and László G Nyúl. Localization of visual codes in the dct domain using deep rectifier neural networks. *International Workshop on Artificial Neural Networks and Intelligent Information Processing: Proceedings of ANNIIP*, pages 37–44, 2014

Péter Bodnár and László G. Nyúl. 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

Péter Bodnár and László G Nyúl. Barcode detection with uniform partitioning and distance transformation. *IASTED International Conference on Computer Graphics and Imaging*, pages 48–53, 2013

Péter Bodnár and László G Nyúl. Improving barcode detection with combination of simple detectors. In *The 8th International Conference on Signal Image Technology (SITIS 2012)*, pages 300–306, 2012

Péter Bodnár and László G. Nyúl. Barcode detection with morphological operations and clustering. In *Signal Processing, Pattern Recognition, and Applications, Proceedings of the Ninth IASTED International Conference on*, pages 51–57, 2012

Other publications

Péter Bodnár, Nyúl László, Tamás Grósz, and László Tóth. Vizuális kódok lokalizációja mély egyenirányított neurális háló használatával. In *A Képfeldolgozók és Alakfelismerők Társaságának 10. országos konferenciája - KÉPAF 2015*, pages 546–561, 2015

Péter Bodnár and László G. Nyúl. QR kód lokalizáció kaszkádolt gyenge osztályozók használatával. In *A Képfeldolgozók és Alakfelismerők Társaságának 10. országos konferenciája - KÉPAF 2015*, pages 712–721, 2015

Péter Bodnár and László G. Nyúl. QR code localization using boosted cascade of weak classifiers. In *The 9th Conference of PhD Students in Computer Science (CSCS 2014): Volume of Extended Abstracts*, pages 6–7, 2014

Péter Bodnár and László G. Nyúl. Vizuális kódok lokalizálásának javítása egyszerű jellemzők kombinációjával. In *A Képfeldolgozók és Alakfelismerők Társaságának 9. országos konferenciája - KÉPAF 2013*, pages 483–495, 2013

Péter Bodnár and László G. Nyúl. Barcode detection with uniform partitioning and morphological operations. In *The 8th Conference of PhD Students in Computer Science (CSCS 2012): Volume of Extended Abstracts*, pages 4–5, 2012

Bibliography

[1] Robert Adelmann. Toolkit for bar code recognition and resolving on camera. In *Phones – Jump Starting the Internet of Things. In: Informatik 2006 workshop on Mobile and Embedded Interactive Systems*, 2006.

[2] D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

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

[4] Péter Bodnár, Tamás Grósz, László Tóth, and László G Nyúl. Localization of visual codes in the dct domain using deep rectifier neural networks. *International Workshop on Artificial Neural Networks and Intelligent Information Processing: Proceedings of ANNIIP*, pages 37–44, 2014.

[5] Péter Bodnár and László G Nyúl. Improving barcode detection with combination of simple detectors. In *The 8th International Conference on Signal Image Technology (SITIS 2012)*, pages 300–306, 2012.

[6] Péter Bodnár and László G Nyúl. Barcode detection with uniform partitioning and distance transformation. *IASTED International Conference on Computer Graphics and Imaging*, pages 48–53, 2013.

[7] Péter Bodnár and László G Nyúl. QR code localization using boosted cascade of weak classifiers. In *Image Analysis and Recognition*, pages 338–345. Springer International Publishing, 2014.

[8] Péter Bodnár and László G Nyúl. Localization of visual codes using fuzzy inference system. In *VISAPP 2015 Proceedings of the 10th International Conference on Computer Vision Theory and Applications*, pages 345–352. SciTePress, 2015.

[9] Péter Bodnár, Nyúl László, Tamás Grósz, and László Tóth. Vizuális kódok lokalizációja mély egyenirányított neurális háló használatával. In *A Képfeldolgozók és Alakfelismerők Társaságának 10. országos konferenciája - KÉPAF 2015*, pages 546–561, 2015.

[10] Péter Bodnár and László G. Nyúl. Barcode detection with morphological operations and clustering. In *Signal Processing, Pattern Recognition, and Applications, Proceedings of the Ninth IASTED International Conference on*, pages 51–57, 2012.

[11] Péter Bodnár and László G. Nyúl. Barcode detection with uniform partitioning and morphological operations. In *The 8th Conference of PhD Students in Computer Science (CSCS 2012): Volume of Extended Abstracts*, pages 4–5, 2012.

[12] Péter Bodnár and László G. Nyúl. Barcode detection using local analysis, mathematical morphology, and clustering. *Acta Cybernetica*, 21:21–35, 2013.

[13] Péter Bodnár and László G. Nyúl. 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.

[14] Péter Bodnár and László G. Nyúl. Vizuális kódok lokalizálásának javítása egyszerű jellemzők kombinációjával. In *A Képfeldolgozók és Alakfelismerők Társaságának 9. országos konferenciája - KÉPAF 2013*, pages 483–495, 2013.

[15] Péter Bodnár and László G. Nyúl. QR code localization using boosted cascade of weak classifiers. In *The 9th Conference of PhD Students in Computer Science (CSCS 2014): Volume of Extended Abstracts*, pages 6–7, 2014.

[16] Péter Bodnár and László G. Nyúl. Improved QR code localization using boosted cascade of weak classifiers. *Acta Cybernetica*, 22:21–33, 2015.

[17] Péter Bodnár and László G. Nyúl. QR kód lokalizáció kaszkádolt gyenge osztályozók használatával. In *A Képfeldolgozók és Alakfelismerők Társaságának 10. országos konferenciája - KÉPAF 2015*, pages 712–721, 2015.

[18] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, 1986.

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

[20] N. Dalal and B. Triggs. 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.

[21] Gustavo De Almeida Neves, Fonseca Eduardo Telmo Santos, Eduardo Manuel De Freitas Jorge, and Alberto Almeida De Azevedo Filho. Automatic system and method for tracking and decoding barcode by means of portable devices having digital cameras, July 5 2012. US Patent App. 12/981,616.

[22] Michael B. Dillencourt, Hannan Samet, and Markku Tamminen. A general approach to connected-component labeling for arbitrary image representations. *J. ACM*, 39:253–280, 1992.

[23] Markéta Dubská, Adam Herout, and Jiří Havel. Real-time precise detection of regular grids and matrix codes. *Journal of Real-Time Image Processing*, pages 1–8, 2013.

[24] Richard L. Dunlap and William A. Slat. Application of radio frequency identification, April 24 2012. US Patent 8,164,457.

[25] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell Computing and Information Science, 2004.

[26] Orazio Gallo and Roberto Manduchi. Image-based barcode reader, January 27 2011. WO Patent App. PCT/US2010/002,023.

[27] Orazio Gallo and Roberto Manduchi. Reading 1D barcodes with mobile phones using deformable templates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(9):1834–1843, 2011.

[28] Jerry Zeyu Gao, Lekshmi Prakash, and Rajini Jagatesan. Understanding 2d-barcode technology and applications in m-commerce-design and implementation of a 2d barcode processing solution. In *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, volume 2, pages 49–56. IEEE, 2007.

[29] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. AISTATS*, pages 249–256, 2010.

[30] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier networks. In *Proc. AISTATS*, pages 315–323, 2011.

[31] Tamás Grósz, Péter Bodnár, László Tóth, and László G Nyúl. QR code localization using deep neural networks. In *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on*, pages 1–6. IEEE, 2014.

[32] Robert M Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, 1979.

[33] Dong-Chen He and Li Wang. Texture features based on texture spectrum. *Pattern Recognition*, 24(5):391–399, 1991.

[34] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[35] Pavel Šimurda. Barcode localization in image. In *Information Sciences and Technologies Bulletin of the ACM Slovakia*, volume 3, pages 55–56, 2011.

[36] Xiaojun Qi James Juett. Barcode localization using bottom-hat filter. *NSF Research Experience for Undergraduates*, 2005.

[37] Eugene Joseph and Theo Pavlidis. Bar code waveform recognition using peak locations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):630–640, 1994.

[38] Melinda Katona and László G. Nyúl. A novel method for accurate and efficient barcode detection with morphological operations. In *The 8th International Conference on Signal Image Technology (SITIS 2012)*, pages 307–314, 2012.

[39] Nahum Kiryati, Yuval Eldar, and Alfred M. Bruckstein. A probabilistic hough transform. *Pattern Recognition*, 24(4):303–316, 1991.

[40] Chuck Kurtz, Gary E. Desjardins, and Stephen J. Sanchez. Self checkout system with automated transportation conveyor, April 17 2007. US Patent 7,204,346.

[41] Yann LeCun, Leon Bottou, Genevieve Orr, and Klaus-Robert Muller. Efficient backprop. In G. Orr and Muller K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998.

[42] Yih-Gong Lee, Jia-Hong Lee, and Yuang-Cheh Hsueh. Texture classification using fuzzy uncertainty texture spectrum. *Neurocomputing*, 20(1):115–122, 1998.

[43] Bernhard Lenk. *Handbuch der Automatischen Identifikation 1: ID-Techniken, 1D-Codes, 2D-Codes, 3D-Codes*. Handbuch der automatischen Identifikation. Lenk Monika Fachbuchverla, 2003.

[44] Rainer Lienhart, Alexander Kuranov, and Vadim Pisarevsky. 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.

[45] Daw-Tung Lin and Chin-Lin Lin. Multi-symbology and multiple 1D/2D barcodes extraction framework. In Kuo-Tien Lee, Wen-Hsiang Tsai, Hong-Yuan Liao, Tsuhan Chen, Jun-Wei Hsieh, and Chien-Cheng Tseng, editors, *Advances in Multimedia Modeling*, volume 6524 of *Lecture Notes in Computer Science*, pages 401–410. Springer Berlin / Heidelberg, 2011.

[46] Daw-Tung Lin and Chin-Lin Lin. Automatic location for multi-symbology and multiple 1D and 2D barcodes. *Journal of Marine Science and Technology*, 21(6):663–668, 2013.

[47] Daw-Tung Lin, Min-Chueh Lin, and Kai-Yung Huang. Real-time automatic recognition of omnidirectional multiple barcodes and dsp implementation. *Machine Vision and Applications*, 22:409–419, 2011.

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

[49] Tony Lindeberg. *Scale-space theory in computer vision*. Springer, 1993.

[50] Christine M. Lowmaster. Blood component container, September 15 2009. US Patent 7,588,193.

[51] Danny J. Montanari and Glenn M. Coleman. Method for tracking the production history of food products, December 26 1995. US Patent 5,478,990.

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

[53] T. Ojala, M. Pietikainen, and D. Harwood. 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.

[54] R. Oktem. Bar code localization in wavelet domain by using binary morphology. In *Signal Processing and Communications Applications Conference, 2004. Proceedings of the IEEE 12th*, pages 499–501, 2004.

[55] Devi Parikh and Gavin Jancke. 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.

[56] Stanley K Peterson. Method of selecting and storing airline ticket data, March 28 2006. US Patent 7,017,806.

[57] Giovanni Ramponi. A cubic unsharp masking technique for contrast enhancement. *Signal Processing*, 67(2):211–222, 1998.

[58] Johann C. Rocholl. *Robust 1D Barcode Recognition on Mobile Devices.* PhD thesis, Institute of Visualization and Interactive Systems, University of Stuttgart, 2010.

[59] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[60] Mehmet Ilhami Safran. A computer vision based barcode reading system. Master's thesis, Atılım University, 2008.

[61] Frank Seide, Gang Li, Xie Chen, and Dong Yu. Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *Proc. ASRU*, pages 24–29, 2011.

[62] Jean Serra. *Image analysis and mathematical morphology.* Academic Press, Inc., 1983.

[63] R. Shams and P. Sadeghi. Bar code recognition in highly distorted and low resolution images. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 1, pages I–737 –I–740, 2007.

[64] Linda G. Shapiro and George C Stockman. *Computer Vision*. Prentence Hall, 2001.

[65] Moti Shniberg, Yaron Nemet, and Erez Sali. Method for automatic identification and data capture, October 5 2004. US Patent 6,801,245.

[66] Stephen P. Shoemaker. Barcode ticket reader, May 11 2004. US Patent 6,732,926.

[67] Gábor Sörös and Christian Flörkemeier. 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.

[68] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[69] Michael J. Swain and Dana H. Ballard. Color indexing. *Int. J. Comput. Vision*, 7(1):11–32, 1991.

[70] István Szentandrási, Adam Herout, and Markéta Dubská. 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.

[71] Ender Tekin and James Coughlan. A Bayesian algorithm for reading 1D barcodes. In *Proceedings of the 2009 Canadian Conference on Computer and Robot Vision*, CRV '09, pages 61–67, Washington, DC, USA, 2009. IEEE Computer Society.

[72] Ender Tekin and James M. Coughlan. An algorithm enabling blind users to find and read barcodes. In *Applications of Computer Vision (WACV), 2009 Workshop on*, pages 1–8, 2009.

[73] Ender Tekin and James M. Coughlan. A mobile phone application enabling visually impaired users to find and read product barcodes. In *Proceedings of the 12th international conference on Computers helping people with special needs*, pages 290–295, Berlin, Heidelberg, 2010. Springer-Verlag.

[74] Blocher J. Thomes, Brinker F. Emil, and Walter W. Sanville. Automatic car identification system, November 24 1970. US Patent 3,543,007.

[75] László Tóth and Tamás Grósz. A comparison of deep neural network training methods for large vocabulary speech recognition. In *Proceedings of TSD*, pages 36–43, 2013.

[76] Timothy R. Tuinstra. Reading barcodes from digital imagery. Technical report, Cedarville University, 2006.

[77] Paul Viola and Michael Jones. 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.

[78] Gregory K. Wallace. The JPEG still picture compression standard. *Consumer Electronics, IEEE Transactions on*, 38(1):xviii–xxxiv, Feb 1992.

[79] Kongqiao Wang, Yanming Zou, and Hao Wang. Bar code reading from images captured by camera phones. In *2005 2nd International Conference on Mobile Technology, Applications and Systems*, page 6, 2005.

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

[81] Sue Wu and Adnan Amin. Automatic thresholding of gray-level using multistage approach. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 493–497 vol.1, 2003.

[82] Sherin M. Youssef and Rana M. Salem. Automated barcode recognition for smart identification and inspection automation. *Expert Systems with Applications*, 33(4):968–977, 2007.

[83] Alessandro Zamberletti, Ignazio Gallo, Simone Albertini, and Lucia Noce. Neural 1D barcode detection using the hough transform. *Information and Media Technologies*, 10(1):157–165, 2015.

[84] Chunhui Zhang, Jian Wang, Shi Han, Mo Yi, and Zhengyou Zhang. Automatic real-time barcode localization in complex scenes. In *Proceedings of International Conference on Image Processing*, pages 497–500, 2006.

[85] Bin Zhou, Shumei Lan, Kai Sun, Jie Cao, Huajun Yu, and Yongliang Chen. Double thresholds with a membership function applied to qr image recognition. In *The Fuzzy Systems, Knowledge Discovery, and Natural Computation Symposium*, pages 59–63, 2013.