Acta Cybernetica **22** (2015) 101–116.

Fast Recognition of Natural Feature Identifiers by a Mobile Phone[*]

Melinda Katona[†] and László G. Nyúl[†]

Abstract

One of the most important and widely used technique for automatic identification is the use of visual codes. Identifiers encoded in various symbols and patterns make electronic reading possible, that greatly helps and speeds up processing, e.g., at cashier lines, warehouse transactions, high speed processing places, production lines. The common codes designed using geometric patterns usually identify types or entities. However, patterns can be produced that, by their nature, are unique and thus can be used to validate originality or authenticity. In this paper, we focus on the automatic localization and recognition of a kind of natural feature identifier (NFI). We present an image processing algorithm that successfully locates NFI code region in an image taken by a mobile camera and extracts features of the NFI glitters that can be the basis for recognition. We also show preliminary experimental results.

Keywords: natural feature identifier, QR code detection, pattern recognition, image processing, mathematical morphology, mobile phone

1 Introduction

Identification is an essential component of our everyday life and our surroundings are full of labels, markers, patterns that identify certain things. A large set of such identifiers are of the visual type, i.e. can be observed via imaging by a camera or a scanner. Artificial identifiers such as serial numbers or alphanumeric identifiers are often attached to products or objects either as text or encoded in a compact graphical representation, such as the traditional barcodes or the more recent QR codes that can encode considerable amount of information. As opposed to codes that follow such artificially created patterns, natural feature identifiers can follow

[*]The research reported here was financed by InterSoft Hungary Ltd. within an R&D project and all rights to commercial use of the resulting technology have been transferred to the sponsoring firm. Research of Melinda Katona was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP 4.2.4.A/2-11-1-2012-0001 'National Excellence Program'.

[†]Department of Image Processing and Computer Graphics, University of Szeged, E-mail: {mkatona,nyul}@inf.u-szeged.hu

DOI: 10.14232/actacyb.22.1.2015.7

rather unique patterns which makes them highly secure and irreproducible. In this sense they are closer to biometrical identifiers, such as fingerprints, iris or retina pictures, that are also unique, basically irreproducible, and inseparable from the objects that they uniquely identify. Labels with printed identifiers can be usually easily reproduced and, depending on the technology, removed and reattached to a different object. Techniques exist for producing non-detachable labels and when combined with natural features, may become a highly secure means for identification.

When coming to the automatic identification/recognition of objects, algorithms are needed to automatically locate and decode the identifiers attached to the objects. A large number of publications deal with locating various codes and patterns in images and recognizing or matching the found pattern, i.e. identifying the object/subject. Very different techniques are required for natural biometric IDs, such as fingerprints [7], iris [3] or retina patterns [1], and for artificial IDs, such as 1D barcodes [13] or QR codes [2, 4, 10, 11]. For the latter type, once the code region is located and the code elements can be read, decoding the information content of graphical codes is rather straightforward. The situation is more complex when the identifier pattern is the result of natural processes and thus can be considered almost random and continuous as opposed to traditional well-structured graphical codes.

In the age of pervasive IT and big data, privacy issues are also becoming increasingly important. De-identification, the process of concealing the identities of individual persons or objects is an important tool for safely analyzing the mass amount of data captured by cameras or other means about our world. In order to achieve proper anonymization or de-identification, one needs to know the possible identifying features, markers, patterns, and techniques are required to automatically recognize such patterns, and to manipulate the data so that the content can still be used for the intended purposes while privacy issues are correctly solved. Thus efficient and reliable automatic visual code localization and recognition is essential both when the aim is to identify objects and when the aim just the opposite, i.e., to conceal their identity.

In this paper, we focus on the automatic localization and recognition of a certain kind of natural feature identifier (NFI) but also give an efficient algorithm to detect QR codes, as the two types of code are combined in the particular application settings we present.

2 Materials and methods

2.1 The NFI label and the task

The identifying label in our case contains a combination of artificial and natural feature identifiers. A standard QR code of a given size is in the center of the label area. This code can encode any content relevant in the application context, e.g. a serial number or a key into a database. Figure 1 shows a schematic drawing

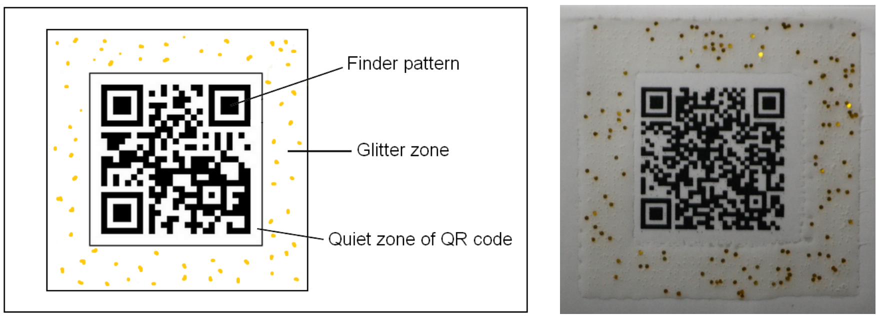


Figure 1: A sketch of the NFI label layout (left) and an image of a real prototype label (right).

of the label layout and an image of a real piece of a prototype label. The QR box is surrounded by a random set of particles or glitters in a window frame of given width and background color. The number, color, size, shape, and position of the glitters can all be considered random variables. Particles and the pattern can be made of various kinds of material by various kinds of technology. In the examples shown in this paper, glitters are made of metallic flakes and are fixed in thin layer of transparent glue or under a transparent foil. Even glitters made by the same technology may reflect light differently which is mostly caused by the small differences in their surface angle w.r.t. the light source and camera position. (In the right image in Fig.1, a couple glitters (in the top-right quadrant) appear very bright, while most of the others are dark, but there are also a few of faint color. Figure 2 shows a few further sample images of prototype labels taken under various conditions.

In the presumed application context each label uniquely identifies a piece of product/parcel/object. After creating/manufacturing the label, it is supposed to be tracked through its lifetime and after it has been attached to an object, the uniquely identified object is tracked as well. Counterfeiting by duplicating the visual appearance of a known valid label would signal the tracking system if an identifier shows up at a place where it is not supposed to be.

In our envisioned setup, a reference photo is taken of each printed label under controlled positioning and lighting condition as part of the production process. (Instead, a few reference photos could also be taken and their information fused to provide better reference features for the matching.) Our task is to locate the label in an image taken by a camera of a mobile phone or tablet, find and decode the QR code in the label, extract the NFI features (glitters and their properties) from the image, and match them with the features extracted from the reference image(s) to reliably identify objects and detect falsified labels. The algorithm is expected to work on images taken by average mobile phone cameras (say, e.g. 5 MPix) under non-perfect conditions (e.g. uneven lighting, shadows, water drops, frost, or other surface artifacts) and feature extraction should also run on a standard mobile

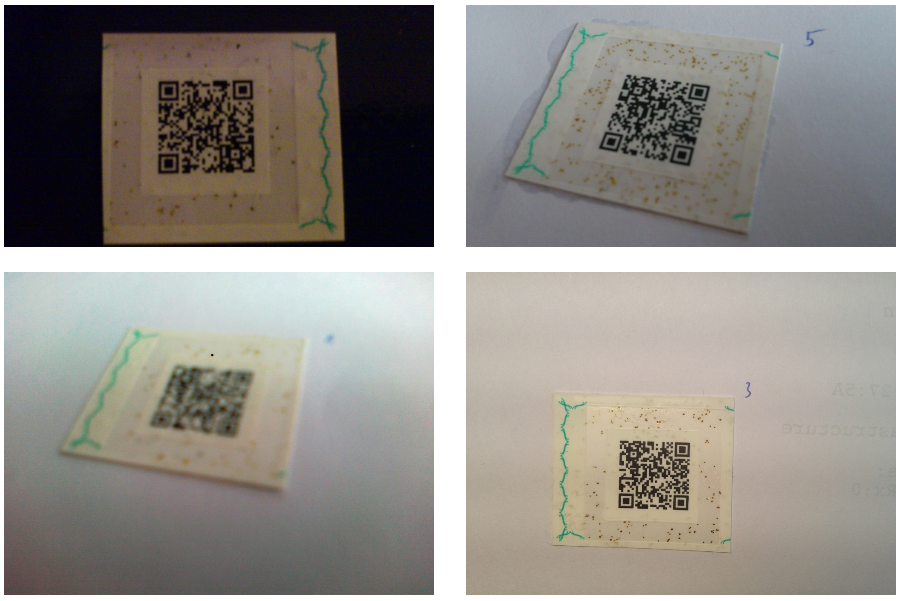


Figure 2: Sample images of prototype NFI labels taken under various conditions. The wavy pattern on one side of the piece is an artifact of the manufacturing has nothing to do with the NFI. (These prototypes are simply not fully finished and cut out from the printing sheet.) For the relevant structure of the label see Fig.1.

phone or tablet. It is also important to note here, that matching/verification is done purely based on the features extracted from the images, without any image transfer between the mobile client and the server infrastructure (neither the reference image nor the one being checked).

The following steps can be identified in the process:

1. image acquisition,
2. image processing,
3. identification.

Standard tools exist for the image acquisition. In this paper, we focus on the image processing part and with the last part we only deal to the level necessary to present the usability of the image processing part.

2.2 Image processing

Depending on the label specification, different image processing approach is required. If the label specification allows a good contrast between the glitter background and the surroundings (e.g. a dark background with bright glitters and labels are applied to white/bright surfaces), the outer frame, the glitter zone can be easily identified based on color contrast (using an appropriately chosen color space dimension), thresholding, and simple morphology. When the label specifications do not use dark glitter background (as in our prototype setup), the bright glitter zone is hard (almost impossible) to differentiate from the white surroundings so we decided to indirectly determine its bounds. Thus, our image processing pipeline consists of the following major steps:

1. locate the QR code (position and orientation),
2. delineate the glitter zone,
3. find the glitters,
4. extract features about the glitter pattern.

The QR code in the middle of the label has a very specific pattern and many algorithms exist to detect QR codes in an image. Unfortunately, most QR reader applications available on the mobile platform work only if the QR code occupies the major part of the field of view, it is well (centrally) positioned in the image, and it is properly oriented. Assistance is sometimes provided in the application to aid the user in finding the position, zoom, and focus and perhaps give some signs (or even more, actually take the picture without requiring the user to hit a button) when everything seems appropriate to take the picture for successful QR code decoding. As we are not only interested in the QR code, but the surrounding glitter zone is also (even more) important, the algorithm must be more robust to image acquisition differences and artifacts.

Since we need an algorithm that works on images coming from a wide range of mobile cameras, we decided to standardize the image size before further image processing takes place. We rescaled the input image to a given width such that the aspect ratio is not changed. The purpose of this size normalization is twofold: 1) to reduce the processing time to allow almost realtime applications, and 2) to design an algorithm that works with a single parameter setting on a wide range of images coming from various mobile cameras. After the initial tests, we settled at 450 pixel wide images that turned out to be sufficiently detailed but at the same time not too large, and basically all mobile cameras have at least such resolution.

The native RGB images (Fig. 3(a)) are converted to the device-independent L*a*b* color space, and in further processing we use the L* (luminosity) channel. In our case L* channel usage turned out to be more effective than other simple weighted grayscale conversions of the RGB space such as the V channel from HSV.

Since input images are distorted by uneven (or even non-sufficient) lighting, we enhance the luminance image by contrast stretching (Fig. 3(b)). This seams a valid step considering the assumption that the QR code elements show high contrast w.r.t. background.

In the following step we use the known fact that QR code is square shaped, and apply grayscale morphological opening with a square shaped structuring element (Fig. 3(c)). This is followed by LoG filtering [5] (Fig. 3(d)) and binarization

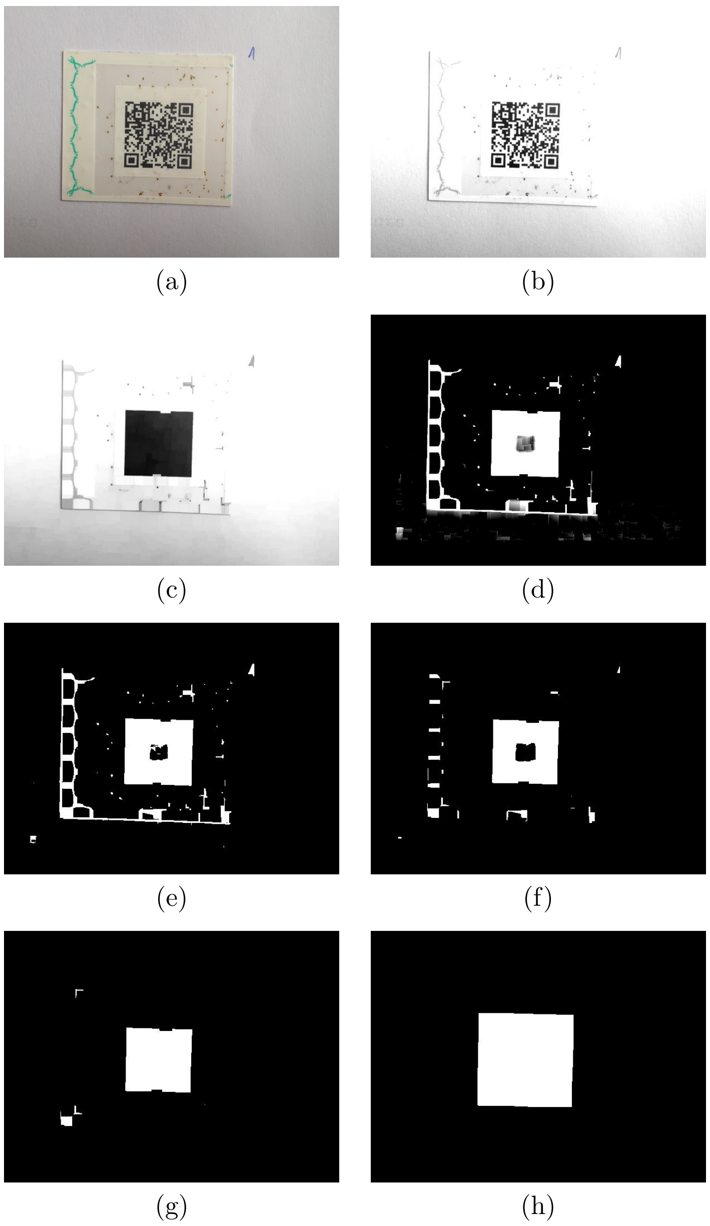


Figure 3: Intermediate stages of the QR code localization pipeline. (a) input image, (b) image after contrast stretching, (c) resulting image after morphological opening, (d) LoG filtered image, (e) binary image after LoG filtering threshold, (f) separate unconnected components with morphological erosion, (g) after eccentricity thresholding, and (h) after major axis length thresholding.

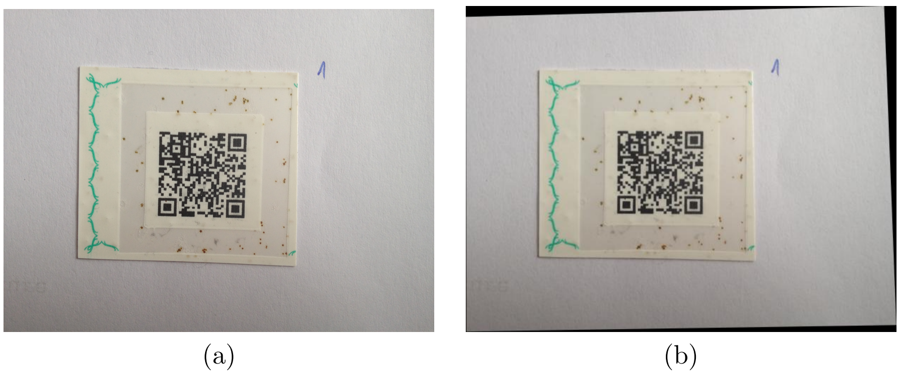


Figure 4: (a) original image, (b) rectified image. The QR code region on the latter appears a well aligned square regardless of the original orientation of the camera.

(Fig. 3(e)). The resulting image so far contains many false positive regions, many small connected components which satisfy the criteria so far but do not belong to barcode regions. A morphological erosion with a smaller structuring element helps removing the majority of these artifacts (Fig. 3(f)). Further filtering is done by using thresholds on eccentricity (Fig. 3(g)) and major axis length (Fig. 3(h)) and in the result only the QR code component remains. A comparison of our QR code detector with some other methods [9, 8] from the literature, applied to a set of synthetic as well as to real images, can be found in [6].

There is a quiet zone around the QR pattern but the detected component does not include this, so we use morphological dilation to increase the object to extend the detected QR code region to include the quiet zone as well. This helps producing a square like component with smoother boundaries (without this the detected component has ragged edges) which does not interfere with the NFI label since the quiet zone separates the QR pattern from the NFI zone.

Once the QR code position is known, orientation can be determined. Remember, we want a robust algorithm that does not depend on proper alignment of the camera when taking the picture. Nevertheless, knowing the orientation of the QR code is required for the QR decoder and also forms the basis for describing the glitter pattern in a comparable way.

The corner points of the extended QR code region are detected as intersection points of the lines fitted to the set of contour points of the segmented mask. [12] The projection transformation is determined based on these corner points assuming that the original pattern is a square and distortions are due to the varying positioning and angles of the camera taking the picture. Then the image can be rectified using the found transformation (Fig. 4).

In addition to rectification, orientation of the QR code needs to be determined, too. Rectification and proper orientation of the image are required to be able to define a standard coordinate system for the glitters. Without this, images taken

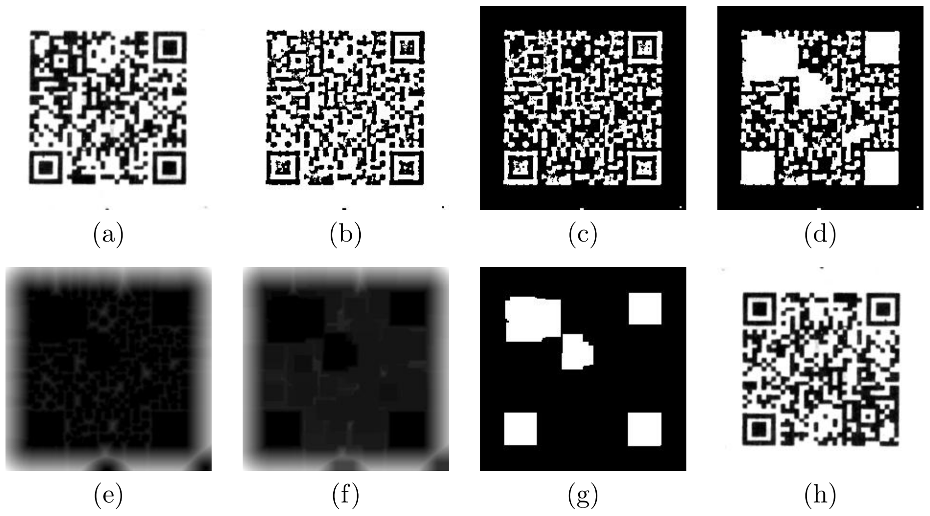


Figure 5: Intermediate steps of QR code orientation detection, from (a) initial QR frame, through (b) binarization, (c) complement, (d) hole filling, (e) distance map, (f) grayscale morphological closing, (g) thresholding, to (h) the reoriented frame.

of the same label by different cameras at different times could not be compared to each other or to the reference. We shall store all extracted features in the reference database with respect to a QR code with standard orientation (see Fig. 1).

The square shaped locator marks in the corners of the QR code can be used to determine proper orientation. To detect the locators we perform the following operations on the segmented QR zone part of the image. The grayscale QR image (Fig. 5(a)) is binarized by adaptive thresholding (Fig. 5(b)) and hole filling (Fig. 5(d)) is applied to the complement (inverted) image (Fig. 5(c)). A distance map is calculated (Fig. 5(e)), grayscale morphological closing is performed on the map (Fig. 5(f)), and the result is thresholded (Fig. 5(g)). The obtained binary segments are further filtered by using a rule-set. Since we now work only on the segmented QR region, several priors can be used. The centroids of the locator marks shall have minimal or maximal position with respect to the others, and one of the locators in a pair along a line has minimal and the other has maximal position.

The above process is based on the assumption that the input image is not distorted to an extent that would prevent the determination of the locator mark positions. This, however, is a valid assumption, since in a real application, QR decoding is also needed and thus a certain level of image quality is required. If the application fails to detect the locators, it may report it to the user and request a new photo to be taken after repositioning the camera.

Given the segmented QR code, the NFI zone (containing the glitters) can be easily determined since the size of the QR code, the quiet zone, as well as the size

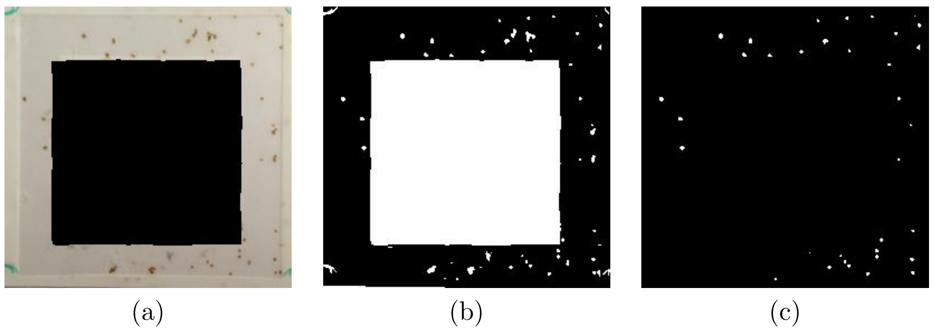


Figure 6: (a) limited glitter zone, (b) adaptive thresholding result, (c) segmented glitters which get into identification

and shape of the NFI zone, and even the colors used in the printing process are defined in the specification. Based on these information about the label geometry, we can create a mask for the glitter zone and restrict glitter detection to this frame (Fig. 6(a)).

The last step of the image processing tasks is the detection of glitters within the NFI zone, for which we use adaptive thresholding (Fig. 6(b)). The wrongly detected objects are filtered out using various morphological features, such as area and circularity (Fig. 6(c)).

Since input images can be of different resolution, QR-NFI labels may appear in various location, size, and orientation in the picture area, it is absolutely necessary to define a normalized geometry in which the detected (glitter) features can be compared between the actual and the reference images. In our prototype setup we defined a standard coordinate system to be centered at the center of the QR code (and the NFI zone frame), aligned according to the zone edges (after rectification and standard QR orientation is applied). The scale is defined so that the glitter zone spans -1.0 to 1.0 both horizontally and vertically. In the later process, the glitter positions, sizes, and other shape features, geometrical distances are all expressed with respect to this standard coordinate system.

2.3 Identification

In this section we outline the process of identification or NFI recognition that can be performed using features extracted in the image processing phase. Each of the following subtasks can be solved in various simple or more sophisticated ways:

1. find matching points,
2. calculate similarity measure,
3. decide about acceptance or rejection.

A wide range of features can be extracted from the images and be used in the above subtasks. Below is a non-exhaustive list of such features:

- number of detected glitters,
- number of matching (paired) glitters (glitter in the image paired with a glitter in the reference),
- number of outlier (not paired) glitters,
- distance between glitters (as pixel regions) or their center points,
- glitters' shape features (area, eccentricity, orientation, shortest axis, longest axis, . . .),
- glitters' appearance features (grayscale/color distribution, histograms, cross-correlation, . . .).

Regardless of an established standard orientation and normalized coordinate system, the extracted features are always loaded with some noise, due to imaging artifacts and segmentation inaccuracies. These must be considered in the algorithms for identification or recognition.

In our prototype implementation we used a very limited set of image-based features, in fact, solely the normalized position of the centroid of the detected glitter. Regardless of the simplicity of this setup, the results reported below, show the feasibility of such NFI technology using mobile applications. Also, we only had a small number of images of a couple of labels for this work. A thorough evaluation is planned to be performed once the label specification is settled, and a larger number of labels and images will be available.

Similarly, we stayed with a very simple matching strategy, using geometric distances between the centroids. For each detected glitter we find a matching pair in the reference image such that we consider a glitter matching if it is the nearest glitter in the reference image and its distance is smaller than a threshold value. We average the pairwise distances between matched glitters and their pairs and accept an NFI if this average distance is below a given threshold. This latter threshold is obviously smaller than the threshold used for matching individual glitters.

Note, that here we only consider the task of validating an NFI label, i.e. to check whether the taken image matches its reference. So, this is not about decoding a particular glitter pattern and finding it in a large database. No specific information content is encoded in the glitters, just the pure natural features coming from the label production. Note also, that validation is done purely using the features extracted from the images, and the reference image is not used directly. The particular reference (i.e. the features extracted from the reference image(s)) can be easily selected for validation using the QR code in the middle of the NFI label.

During the parameter setting phase the input images were compared with all images in the small reference database and the thresholds were set to maximize the true acceptance rate while minimizing false detection. In the end, the first threshold (for paring up individual glitters) were set to 0.025 and the label acceptance threshold was set to 0.0065 (both with respect to the standard coordinate system).

3 Evaluation

3.1 Test suite, test environment

The prototype NFI label specification was the following. In an image there are two concentric and aligned square areas. The sides of the outer square are 18 mm and those of the inner square are 12 mm. The inner square is filled with a QR code in black-and-white. The outer frame is monochrome, but there are 30-60 particles made from light-reflective material (glitters), with a maximum size of 400 micron each. The glitter locations are random within the window. (See sample images in Fig. 2).

For testing, we had 80 images captured by 6 different cameras, so image quality (including resolution, color balance, geometric distortions, and lighting conditions) varied considerably.

We implemented the methods in MATLAB, with the help of the Image Processing Toolbox. The results reported in the following tables were obtained on a computer with Intel(R) Core(TM) i7-4700MQ 2.40 GHz CPU.

3.2 Results and discussion

In this section we show results for images taken by an iPhone and a Sony Xperia mobile phone. The reference images were taken by a Canon camera and the same references were used for both mobile sets. Table. 1 shows basic settings of the cameras.

Table 1: Type and image settings of the devices used for the evaluation.

	Canon	iPhone	Xperia
Make	Canon	Apple	Sony Ericsson
Model	PowerShot SX220 HS	iPhone 4S	LT15i
ISOSpeedRatings	1600	50	800
Flash	Flash not fired	Flash not fired	Flash not fired
ExposureMode	Auto	Auto	Auto
White Balance	Auto	Auto	Auto
Image size	4000 x 3000	3264 x 2448	1920 x 1080

In Table 2, the sequential numbers in the column and row labels correspond to individual NFI labels. Elements in the major axis of tables correspond to true matches, i.e. comparison of two images taken by different cameras of the same

NFI label. The values shown in the table are the average pairwise distances of the matched glitters. The values for image pairs that the algorithm accepted are shown in various boldface type and for rejected pairs are shown in normal typeface. Those values marked ***boldface italic*** are true positives (real pairs accepted by the algorithm) and those marked in **<u>boldface underlined</u>** are false positives (non-pairs accepted by the algorithm as a valid pair). In this rather small set of images the other type of error (false negatives, i.e. a true pair that is rejected) did not occur. In this sample the algorithm indicated only 1 mistaken identities for each camera set and 5 (all) true matches.

Table 2: Matching scores for 5 labels using the reference images (taken by a Canon camera) and images taken by two different cameras (iPhone (top table), Xperia (bottom table)). True positive matches are typeset in ***boldface italic*** (in the diagonal), false positives are shown in **<u>boldface underlined</u>** font, and the rest is correctly considered non-matches under the used parameter settings.

	iPhone_1	iPhone_2	iPhone_3	iPhone_4	iPhone_5
Canon_1	***0.0065***	0.0073	0.0075	0.0099	0.0066
Canon_2	0.0099	***0.0017***	0.0066	0.0080	0.0067
Canon_3	0.0093	0.0078	***0.0042***	0.0077	0.0066
Canon_4	0.0086	**<u>0.0057</u>**	0.0073	***0.0034***	0.0067
Canon_5	0.0093	0.0079	0.0072	0.0093	***0.0054***

	Xperia_1	Xperia_2	Xperia_3	Xperia_4	Xperia_5
Canon_1	***0.0062***	0.0076	0.0069	0.0083	0.0096
Canon_2	0.0112	***0.0027***	0.0086	0.0096	**<u>0.0051</u>**
Canon_3	0.0097	0.0069	***0.0045***	0.0105	0.0072
Canon_4	0.0094	**<u>0.0062</u>**	0.0087	***0.0058***	**<u>0.0065</u>**
Canon_5	0.0100	0.0083	0.0072	0.0097	***0.0043***

The computing times are shown in Table 3. Computing time is dominated by the image processing operations and since matching is only to be done to a particular reference image, the identification time is basically independent of the size of the reference database.

Table 3: Computing time statistics (mean, 95% confidence interval)

Operation	Computing time (sec/image)	
	Mean	95% CI
Image processing	1.1129	$[1.0915, 1.1343]$
Verification	3.0×10^{-4}	$[2.24, 3.76] \times 10^{-4}$
Image processing + verification	1.1133	$[1.0918, 1.1347]$

Figure 7 shows image pairs and the relative positions of glitters plotted in the same reference coordinate system.

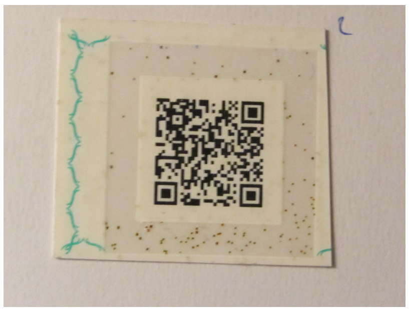
3.3 Mobile application

We implemented a simple demo NFI reader as an Android application for mobile phones and tablets. Image processing was implemented using the openCV4Android SDK. The application can use the built-in camera app (Fig. 8(a)) to take a picture of a scene (or for demonstration purposes it can load a picture taken offline). QR code is detected and decoded, NFI glitters are detected and the NFI pattern is matched against the reference. The application computes a match score under about 1 second. The label is accepted as valid if the score is higher than a given threshold, otherwise the system raises an alarm about an invalid label (Fig. 8(b)). Please note, that images are not transferred, and reference images are not used for validation, only the extracted features.

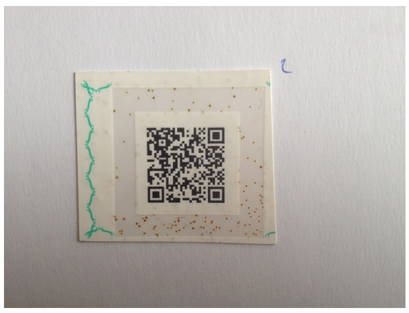
The observant reader may have noticed that the time it took to validate a label using a mobile phone is shorter than that measured on a fast PC using the prototyping MATLAB environment. This is partially due to the fact that when porting the algorithm to the Android platform, some functions used in the MATLAB code had to be replaced by versions available in the openCV library and sometimes on the mobile platform simple, direct functions were used for a simple step, while in the rapid prototyping MATLAB program more complex functions (e.g. 'regionprops') were used, that obviously increased the time with unnecessary computations. Nevertheless, we chose to do the statistical evaluation in the algorithm prototyping MATLAB environment on the PC, since it was easier to handle all images from various sources, all pairings, and the entire evaluation setup and computation. The individual tests on the mobile platform proved to meet the time requirement (validate under 1.4 second) set out beforehand.

4 Conclusion

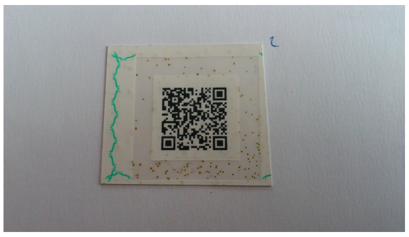
We have presented an image processing algorithm to implement natural feature identifier recognition. The method is robust against variations in image quality, resolution, lighting, and positioning. It automatically detects the QR code, determines the transformation to normalize the QR-NFI label, and detects the NFI glitters within the NFI zone. We also demonstrated, using a simple matching strategy, that such identification is feasible using mobile devices with a recognition time around 1 second.



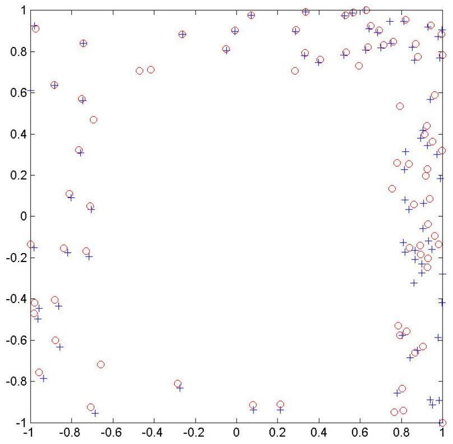
(a)



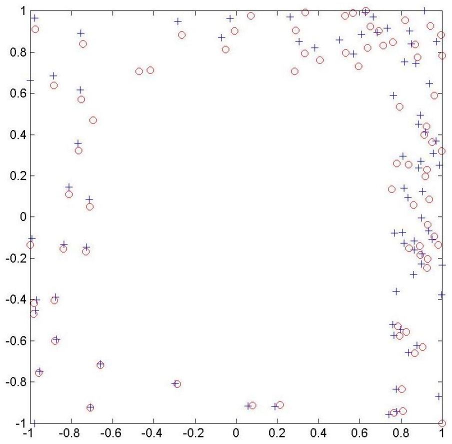
(b)



(c)

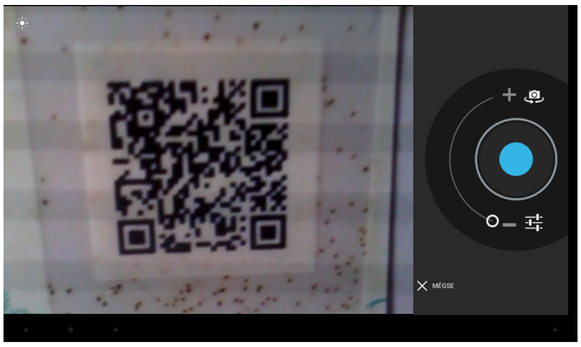


(d)

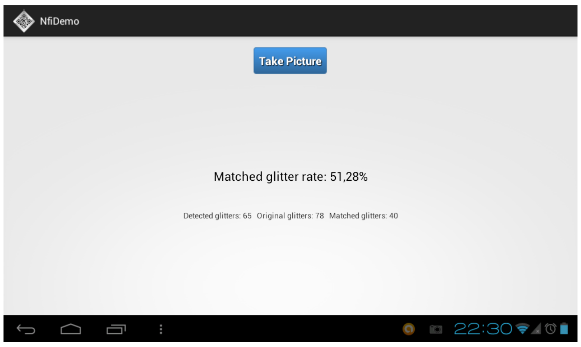


(e)

Figure 7: Images of the same labels: (a) sample (Canon), (b) test (iPhone), (c) test (Xperia). Detected glitter positions in the normalized coordinate system: (d) Canon & iPhone, (e) Canon & Xperia.



(a)



Matched glitter rate: 51,28%

Detected glitters: 65 Original glitters: 78 Matched glitters: 40

(b)

Figure 8: Screens of the demo NFI application. (a) the image acquisition app, (b) the simple results screen.

References

[1] Barkhoda, Wafa, Akhlaqian, Fardin, Amiri, MehranDeljavan, and Nouroozzadeh, MohammadSadeq. Retina Identification Based on the Pattern of Blood Vessels Using Fuzzy Logic. *EURASIP Journal on Advances in Signal Processing*, pages 1–8, 2011.

[2] Belussi, L.F.F. and Hirata, N. S T. Fast QR Code Detection in Arbitrarily Acquired Images. In *Graphics, Patterns and Images (Sibgrapi), 2011 24th SIBGRAPI Conference on*, pages 281–288, 2011.

[3] Chen, Wen-Shiung, Chih, Kun-Huei, Shih, Sheng-Wen, and Hsieh, Chih-Ming. Personal Identification Technique Based on Human Iris Recognition with Wavelet Transform. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, pages 949–952, 2005.

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

[5] Gonzalez, Rafael C. and Woods, Richard E. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., 2006.

[6] Katona, Melinda and Nyúl, László G. Vonalkódok és természetes azonosítók felismerése [Recognizing barcodes and natural identifiers]. In *A Képfeldolgozók és Alakfelismerők Társaságának 10. országos konferenciája - KÉPAF 2015*, pages 562–577, 2015.

[7] Khan, Muhammad Khurram and Zhang, Jiashu. Multimodal face and fingerprint biometrics authentication on space-limited tokens. *Neurocomputing*, 71:3026–3031, 2008.

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

[9] Ohbuchi, Eisaku, Hanaizumi, Hiroshi, and Hock, Lim Ah. Barcode readers using the camera device in mobile phones. In *Proceedings of the 2004 International Conference on Cyberworlds*, CW '04, pages 260–265, 2004.

[10] Sörös, Gábor and Flörkemeier, Christian. Blur-resistant Joint 1D and 2D Barcode Localization for Smartphones. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, pages 1–8, 2013.

[11] Szentandrási, István, Herout, Adam, and Dubská, Markéta. Fast Detection and Recognition of QR Codes in High-resolution Images. In *Proceedings of the 28th Spring Conference on Computer Graphics*, pages 129–136, 2013.

[12] Varjas, Viktor and Tanács, Attila. Car recognition from frontal images in mobile environment. In *Image and Signal Processing and Analysis (ISPA), 2013 8th International Symposium on*, pages 819–823, 2013.

[13] Wachenfeld, Steffen, Terlunen, Sebastian, and Jiang, Xiaoyi. Robust recognition of 1-D barcodes using camera phones. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4, 2008.