Blur-resistant joint 1D and 2D barcode localization for smartphones

Gábor Sörös
Institute for Pervasive Computing
ETH Zurich
Universitaetstrasse 6
Zurich, Switzerland
soeroesg@inf.ethz.ch

Christian Flörkemeier
Institute for Pervasive Computing
ETH Zurich
Universitaetstrasse 6
Zurich, Switzerland
floerkem@inf.ethz.ch

ABSTRACT
With the proliferation of built-in cameras barcode scanning on smartphones has become widespread in both consumer and enterprise domains. To avoid making the user precisely align the barcode at a dedicated position and angle in the camera image, barcode localization algorithms are necessary that quickly scan the image for possible barcode locations and pass those to the actual barcode decoder. In this paper, we present a barcode localization approach that is orientation, scale, and symbology (1D and 2D) invariant and shows better blur invariance than existing approaches while it operates in real time on a smartphone. Previous approaches focused on selected aspects such as orientation invariance and speed for 1D codes or scale invariance for 2D codes. Our combined method relies on the structure matrix and the saturation from the HSV color system. The comparison with three other real-time barcode localization algorithms shows that our approach outperforms the state of the art with respect to symbology and blur invariance at the expense of a reduced speed.

Categories and Subject Descriptors
J.7 [**Computers in Other Systems**]: Consumer products; I.5.4 [**Pattern Recognition**]: Applications

General Terms
Algorithms

Keywords
visual codes, barcode, localization, pattern recognition

1. INTRODUCTION
Barcode scanning with smartphones is becoming popular thanks to a rich variety of e-commerce services. While barcode scanning in the past required dedicated laser scanner or imaging devices, consumers are today using their smartphones to scan barcodes on products to conveniently compare prices and to check product reviews. In enterprise applications, employees are using their smartphones to scan barcodes to reorder products and to track assets.

MUM'13, December 02 - 05 2013, Lulea, Sweden
Copyright 2013 ACM 978-1-4503-2648-3/13/12...$15.00.
http://dx.doi.org/10.1145/2541831.2541844

The smartphone-based barcode scanning pipeline consists of several steps. In the initial step, video frames are acquired from the camera. The frame rate is usually between 5 and 30 frames per second and the resolution in video mode is typically between 640x480 and 1280x720 pixels depending on the required quality and speed of processing. In practice, the scanning algorithms must be limited to a small search window in the image because (blurry) decoding is computationally complex. Due to the lack of a laser beam the users do not see where the scanner is currently reading and they often struggle with continuously aligning the barcode at a dedicated position and angle within the search window. To maximize ease of use, barcodes need to be scanned in the entire camera image. For that a quick barcode localization algorithm is required that selects possible barcode candidates before a particular scanline through the code is decoded using approaches such as ZXing, RedLaser, or Scandit[1].

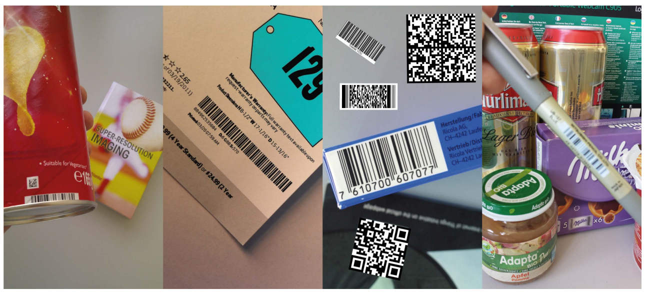

Figure 1: Barcode localization in digital images poses several challenges: different scales, different orientations, different symbologies, blur

The ideal barcode localization algorithm is scale, orientation, barcode-symbology and blur-invariant (see Figure 1) and operates in real-time on the entire camera image. Scale-invariance is important because barcodes are printed in different form factors and the distance to the smartphone camera can vary. Orientation invariance is necessary because the

[1]`http://code.google.com/p/zxing/`
`http://www.redlaser.com`
`http://www.scandit.com`

user will not always place the barcode exactly horizontally in the camera image. There are also a number of different barcode symbologies that need to be successfully detected by the barcode localization algorithm. This includes linear 1D barcodes such as those found on consumer products and on logistical labels (e.g. EAN13, UPC12, CODE39, CODE128) and 2D codes such as the popular QR codes, but also DataMatrix and PDF417 codes. Enterprise applications may require scanning 1D and 2D symbologies in the same algorithm without explicit switching by the user. Since the latest commercial barcode decoders can also decode blurry barcodes, it is also important to localize blurry barcodes in camera images. Blurry barcode images can result from a fixed focus camera, a barcode being placed too close to the camera or the camera focusing on the distant background when the barcode is small (cf. Figure 1).

Previous work focused only on selected aspects of the requirements mentioned above such as orientation invariance and speed for 1D codes or scale invariance for 2D codes. The contribution of this paper is a combined barcode localization approach that aims to address both orientation, scale, symbology and blur invariance. Our results with publicly available data sets show that the method provides robust orientation and scale invariance, and can localize both 1D and 2D barcodes in sharp and blurry images.

The remainder of the paper is organized as follows. In Section 2, we review the related work also including the details of three techniques that represent the state of the art in real-time barcode localization. Section 3 is devoted to the description of our approach while Section 4 presents a comparison of our approach with the three previous ones. We conclude and discuss future work in Section 5.

2. RELATED WORK

Despite the fact that barcodes have an apparent structure for the human eye, the fast and robust localization in digital images is still an active research area. The different approaches presented previously can be categorized by the image processing technique used. Previous work on 1D localization includes simple image filters, orientation histograms, line detection approaches, morphology operators, Gabor filters and harmonic analysis. 2D localization has been previously implemented via image thresholding and scanning, via orientation maps, or via combinations of line/corner/square detection.

2.1 Localization of 1D Barcodes

Simple image filters combine various low-level image features such as image gradients, intensity variance, etc., and try to find bar-like structures [4, 13, 16, 6]. They offer a good compromise when accuracy is not crucial but localization speed is an important aspect in the application. Ando and Hontani [4] detected sharp 1D barcodes by classifying image regions to plain, uni-directional and omni-directional areas and by inspecting the transitions between those areas. Gallo et al. [6] presented a very fast barcode localization technique that despite its simplicity outperforms many complex methods. However, it works only with a single sharp code that lies horizontal (less than 30° rotated) in the image. For each pixel, the algorithm calculates the measure $m = |I_x| - |I_y|$ where I_x and I_y stand for image derivatives in x and y directions, respectively, and $|.|$ means absolute value. m is then blurred with a box filter and binarized by Otsu's method. The idea behind this approach is that horizontally oriented barcodes have strong gradients in the x direction but no gradients in the y direction, so m is very high at bars in a barcode. The box filtering connects the bars into a region. To find the axis-aligned bounding box of the code, Gallo et al. first find the maximum value in m (assuming this is within the barcode region) and search in four directions on the binarized image for the region boundaries. Finally, a rectangle is fitted on the four endpoints. We also apply some of these ideas in our approach.

Orientation histograms are used for 1D barcode localization by Tekin and Coughlan [12]. Their algorithm is part of the BLaDE barcode scanning aid for visually impaired users who have difficulties with aligning codes in a search window. The algorithm builds a histogram of gradient orientations for each 20x20 patch in the image, calculates the entropy (peakedness) of the histogram, and finds its dominant orientations. The patches are then clustered according to their orientation. The resulting regions are tested for alternating edges in their dominant orientation and the most likely barcode candidate is selected. This approach provides orientation invariance but the patch size has to be tuned carefully to the expected code scales. Also many other algorithms look at the distribution of the gradient orientations over image patches; sharp 1D codes have a single peak and 2D codes have two peaks 90 degrees apart. However, when the image gets blurred, many of the thin bars disappear and the code breaks into many disconnected parts. The blur also flattens the orientation histogram of a 2D code, it "smears" the gradients in practically all directions making the common localization algorithms fail.

Line detection with Hough transform has also been used to estimate the orientation of a single barcode close to the camera with little background clutter [1]. Dubska et al. [5] presented a fast localization algorithm that successfully distinguishes 2D codes from text. However, to detect a sufficiently large number of lines, the code must be fairly prominent in the camera image.

Mathematical morphology is another tool for detecting bar structures. Combinations of image erosions and dilations enhance barcode areas but these algorithms tend to produce frequent false positives and will always be constrained by the proper choice of the structuring elements. Detection at different scales requires a search with multiple structuring elements which can be a rather slow process. An overview on morphological localization can be found in the work of Katona et al. [9]. The authors also propose an alternative approach that uses bottom-hat filtering and a distance map and achieve over 90% detection rates in their simulations. Their method can be extended to combine 1D and 2D localization but it often classifies text areas as part of a code.

Gabor filters are combinations of different Gaussian and Sinusoid functions that model the edge-sensitivity of the human primary visual system. They can be used to detect stripe patterns at any orientation and scale. Concrete examples in localization include [11, 14]. These methods offer scale and rotation invariance but tend to be slow.

Harmonic analysis approaches transform the image into the Fourier, DCT, or Wavelet domain and look for barcode-specific features. Examples include [10, 7]. We consider these methods to be too complex for real-time barcode scanning on smartphones.

2.2 Localization of 2D Barcodes

2D-specific methods for codes like DataMatrix, QR, etc., rely on the codes' apparent black and white rectangular structures and special finder patterns with given black-white signal ratios. The typical localization pipeline consists of image thresholding and line-by-line search in the binary image for the finder patterns. Obviously, if the finder patterns are degraded by blur the black-white proportions do not follow the standards anymore.

Alfthan's work [2] is devoted to localizing QR codes in blurry images with three different approaches. First, the author trains an SVM on color and intensity variance features extracted from QR codes. This approach can successfully separate the code area from text but is not scale-invariant. The second, morphology-based method requires precise parameter tuning. Third, the author tries to find the white square support of the code in Hough-space, which is inherently influenced by the success of the underlying edge detection.

Xu et al. [15] presented the first method to localize and also deblur linearly blurred 2D codes using a special camera. They localize the blurry code based on intensity variance, Harris corner density [8] and background substraction. The authors rely on the fact that corners are less sensitive to motion blur than edges so the many corners of a 2D code can still be found in a blurry image.

In summary, there is a vast amount of literature on localizing barcodes in digital images, but none of the existing approaches focuses on all four identified challenges. We developed a combined approach that addresses both orientation, scale, symbology and blur invariance.

3. OUR APPROACH

In this section we present a new localization algorithm for 1D and 2D codes that also works with blurry images. We rely on the facts that 1D codes consist of black and white parallel bars while 2D codes have a black and white grid structure. The grid corners of 2D codes are less sensitive to blur and stay apparent also in blurry images. In our combined algorithm we search for areas with high concentration of edge structures as well as for areas with high concentration of corner structures. We derive two separate barcodeness maps from the structure matrix for 1D and 2D codes. The two maps can be calculated at the same time very efficiently. To further speed up the sharp localization and allow blurry localization we also rely on information from the HSV (hue,saturation,value) color channels that - to the best of our knowledge - has not been proposed before.

3.1 Structure Matrix

We derive our barcodeness measures from the structure matrix $M = \begin{pmatrix} C_{xx} & C_{xy} \\ C_{xy} & C_{yy} \end{pmatrix}$ at each pixel p. The C_{ij} entries of M are calculated from the image derivatives I_x and I_y over an image patch D around the pixel p using a window w:

$$C_{i,j} = \sum_{(x,y)\in D} w(x,y) I_i(x,y) I_j(x,y)$$

As we will see, using the structure information instead of simply the gradient like other methods can enhance also the robustness against blur. The following steps are illustrated in Figure 2.

3.2 Edge and Corner Maps

From the structure matrix, according to the work of Ando [3], we can define the measure m_1 which is strong at edge structures in any orientation and the measure m_2 which is strong at corners. The derivation of these formulas can be found in the Appendix.

$$m_1 = \frac{(C_{xx} - C_{yy})^2 + 4C_{xy}^2}{(C_{xx} + C_{yy})^2 + \epsilon}, \quad m_2 = \frac{4(C_{xx}C_{yy} - C_{xy}^2)}{(C_{xx} + C_{yy})^2 + \epsilon}$$

Next, we blur the two maps by a block filter to connect areas where there is high line density or high corner density. Note that while the applied edge/corner detection is invariant to illumination changes and to rotation, it is not invariant to scale changes. Scale invariance could be added by repeating the calculation of M over bigger and bigger image patches D and selecting extrema over the scales. Fortunately, visual codes contain plenty of edge and corner structures of different size so independent of the distance between the code and the camera, there will be lots of edges and corners detected even using a fix neighborhood size $D = 7$. We achieve good scale invariance by connecting those edges/corners via a big block filter. In our experiments we used a separable box filter of size 30 pixels. This not only connects high-density areas but also removes separated edges/corners of background clutter. There is, however, often text in the vicinity of visual codes that also exhibits high edge and corner density, so a further step is necessary to remove those areas.

3.3 Barcode Saliency Maps

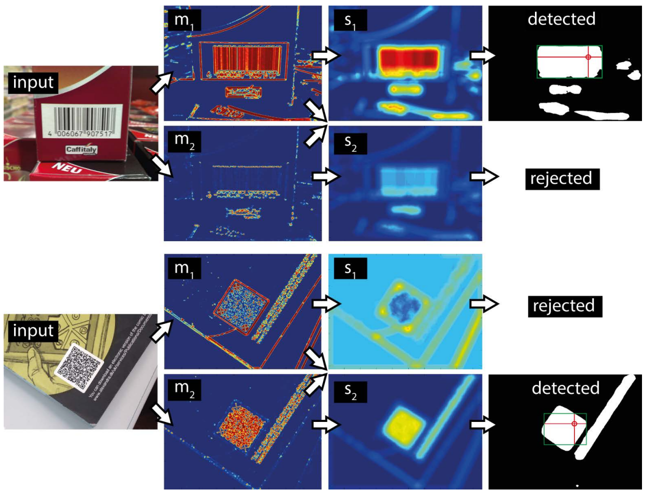

Figure 2: Algorithm outline: m_1: edge density map, m_2: corner density map, s_1: 1D saliency, s_2: 2D saliency, bounding box detection. The color coding is normalized to [0..1] in all figures. (Please zoom in for better viewing.)

The two box-filtered maps are linearly combined to get our final barcode saliency maps s_1 and s_2. The idea behind this is that 1D codes must not contain corner areas (note that this is how text gets removed from 1D codes) and sharp 2D codes must also contain edge areas. The resulting "barcodeness" images are thresholded and barcode borders are found by tracing the binary image in four directions starting from the pixel with maximal strength (cf. Gallo's method [6]). Note that using a more sophisticated method for finding the peak areas in the barcodeness map would allow for detecting multiple codes even with different symbologies.

The above described method works reliably with sharp 1D, sharp 2D, and blurry 2D codes, however, blurry 1D codes are challenging as the blur leaves little structure information in the code area (see Figure 3). If the camera focuses on the background that contains strong parallel lines, the algorithm finds that area more likely to be a barcode. To overcome this limitation we propose an extension to our algorithm using information from the saturation channel.

3.4 HSV Color Information

So far we have not considered that barcodes are almost always printed black and white and in most cases also in a rectangular white support. This is a very important clue because if we look at a sharp and a blurry code in the HSV (hue, saturation, value) color system, their saturation values are very low in both cases. Although blur distorts the intensity structure of the code, smearing black and white together still results in gray. In very blurry cases where only little structure information is left, code localization may be still possible by finding rectangles in the saturation channel (see Figure 3). If both m_1 and m_2 are very low, our algorithm switches to rectangle detection mode. Of course, this works only with the mentioned rectangular white support but it is the case with lots of consumer products. Furthermore, looking at the saturation of each pixel can be a fast first test to reject non-barcode pixels in general. We also augmented our edge/corner detector with a color conversion module and we set all pixels with saturation above 0.25 to zero in both m_1 and m_2 to remove background clutter.

4. EVALUATION

4.1 Implementation

We have developed our algorithm in Matlab and ported it to C++ using the OpenCV framework. The native OpenCV code can also be executed on iOS and Android with little modifications. As speed is crucial in mobile barcode scanning, we compare our method to three others (Gallo2011 [6], Tekin2012 [12], Katona2013 [9]) that represent the state of the art and allow real-time operation on smartphones. To only compare the quality of the saliency maps of the different algorithms, we apply Gallo's method (described in Section 2) in each algorithm to generate a bounding box from the saliency map except for Tekin2012 that already returns a single scanline.

4.2 Test Environment

Mobile barcode scanner applications read video preview images with high frame rate rather than still images to speed up the scanning. Recent smartphone cameras are able to deliver preview frames in 720x960 resolution, so we focus on this image size in all our tests.

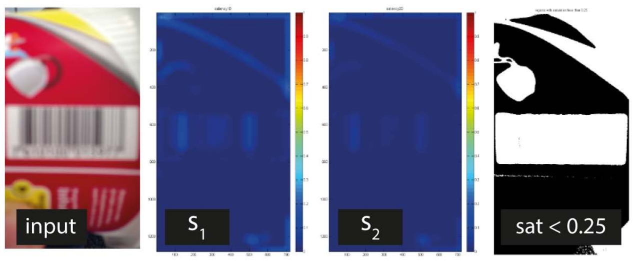

Figure 3: A very blurry 1D code results in low 1D saliency (s_1) and low 2D saliency (s_2) but if wee look at areas with saturation below 0.25, the code rectangle is clearly visible.

We test the 1D performance on the Muenster BarcodeDB[2] data set by Wachenfeld et al. [13]. We did not include their localization approach in our comparison because they assume the user already positioned the code to the center of the image. The algorithms we test do not need to make this assumption. The 1050 images were taken with a Nokia N95 phone with autofocus (AF). Additionally, we also recorded 200 blurry images with an iPhone5 with its AF turned off.

For tests with 2D codes we use the QR code dataset of Dubská et al.[3] which consists of about 400 images with perspective distortions, illumination variations, blur, and surrounding text. The images were taken by a mobile phone in (high-resolution) photo mode. We also recorded 120 new images of QR codes with an iPhone4S in video mode with and without AF. The images contain codes in various scale, orientation and blurriness. Our images are also publicly available[4].

We hand-labeled the code corners in all images and stored the bounding boxes as ground truth. We compared the accuracy of the four algorithms by measuring the overlap of their output with the ground truth. We calculate the Jaccard coefficient $J(A, B) = |A \cap B|/|A \cup B|$ as the overlap measure where A is the of the ground truth bounding box and B is the detected bounding box. A coefficient above 0.5 represents a visually good match when the returned bounding box is bigger than the ground truth, but actually a smaller coefficient would also be acceptable as decoders can deal with codes that cover one third of the search window. When the returned bounding box is smaller than the ground truth we accept only 5% loss. Because the algorithm of Tekin returns a single scanline instead of a bounding box, we calculate its Jaccard coefficient in one dimension along the code axis.

4.3 Experimental Results

4.3.1 1D Performance

In our first test we took 1000 original sharp images from the Muenster dataset and looked at the average Jaccard coffecient of the four algorithms (see Table 1). Our algorithm

[2]`http://cvpr.uni-muenster.de/research/barcode/`
[3]`http://medusa.fit.vutbr.cz/pclines/?p=86`
[4]`http://people.inf.ethz.ch/soeroesg/`

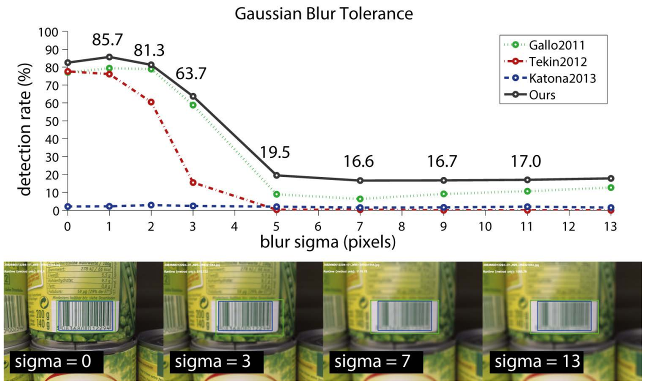

Figure 4: Successful 1D detection rates ($J \geq 0.5$) on 1000 images from the Muenster data set with various degrees of artificial Gaussian blur. The bottom row gives an impression on the effect of the blur parameter sigma.

achieves an average overlap well above 50% while its standard deviation is also lower than that of Gallo2011.

In our second test we investigated the robustness of the algorithm against Gaussian (defocus) blur. We added artificial Gaussian blur to the original images with standard deviation $\sigma = 1, 2, 5, 7, 9, 11, 13$ pixels. Our results are summarized in Figure 4 and example images are listed in Figure 8. With sharp images, our algorithm accurately detects 82.5% of the barcodes out of 1000 images. False positives are mostly caused by a very dominant sharp edge in the background or selecting the wrong code among multiple ones (we always labeled the one which is closest to the image center). False negatives are caused by colorful codes and non-rectangular codes. Our algorithm detected more blurry codes accurately than others.

It is important to note that on sharp images we achieve about the same detection rate as the other algorithms but we make no assumptions about code size (like Katona2013) nor code orientation (like Gallo2011) nor code position (like Wachenfeld2010). Although Katona2013 [9] reports over 90% detection rate in simulations, the method performs weakly in our tests. It almost always returns a bounding box that is too big and contains text and other objects around the code. Gallo2011 is fast and works well even with fairly blurry codes, but often returns a too small bounding box because a blurry code breaks into many parts. Its high detection rate is thanks to the fact that most codes in the data set are horizontal. Tekin2012 works with multiple orientations, however, on blurry images it tends to give too short scanlines and returns many false positives.

In our third test we added artificial motion blur to the

Algorithm	Gallo	Tekin	Katona	Ours
Avg. J. coeff. $\overline{J}$	70.89%	81.22%	19.94%	66.47%
Std. dev.	35.42%	25.62%	11.56%	27.77%

Table 1: Average Jaccard coefficients on 1000 sharp images. We defined every $J \geq 0.5$ a good match. (Recall that for Tekin2012 J is calculated only in 1D therefore results in better overlap).

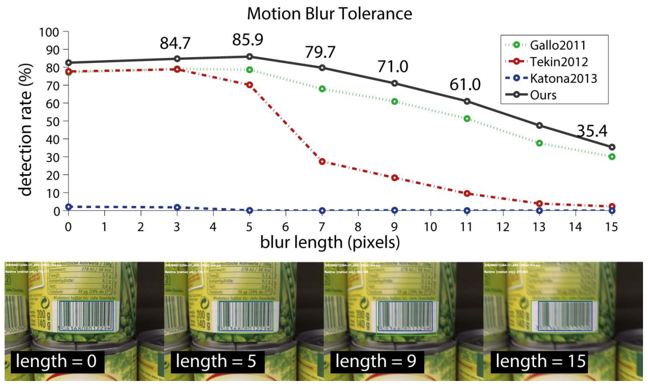

Figure 5: Successful 1D detection rates ($J \geq 0.5$) on 1000 images from the Muenster data set with various degrees of artificial motion blur. The bottom row gives an impression on the effect of the blur length.

1000 sharp images. We chose motion blurs with length $3, 5, 7, 9, 11, 13, 15$ pixels in $135\,^\circ$ direction to make sure we destroy some of the bars. Figure 5 shows that algorithm is more robust to motion blur than the others.

The detection rates on the images we recorded with real defocus and motion blur are 5.2% (Gallo2011), 0.0% (Tekin2012), 8.1% (Katona2013), and 29.8% (ours). Example images are shown in Figure 9. Our method has difficulties with blurry codes without a rectangular frame and sometimes sharp gray stripes in the background are found more likely to be a barcode than the actual blurry one. We can still conclude that our algorithm outperforms the previous 1D localization approaches in blur resistance while it is also scale and rotation invariant and can detect 2D codes as well.

4.3.2 2D Performance

We tested the 2D performance of the algorithm on the QR code dataset of Dubská et al. While their line detection algorithm performs well on big sharp codes of the dataset, our algorithm returns oversized bounding boxes also including the surrounding text. However, on the motion blurred images of the dataset, the corner measure is more reliable than line detection (see Figure 6). Due to our oversized bounding

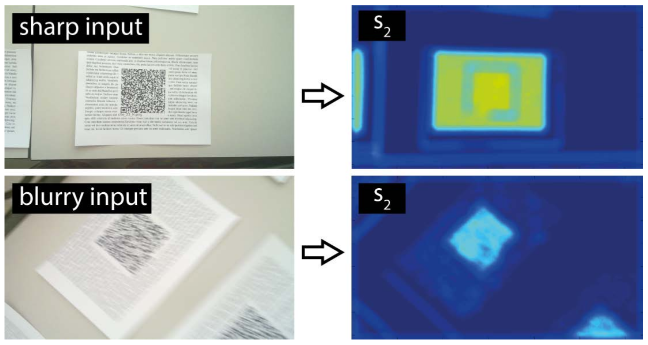

Figure 6: Sharp text around the QR code misleads our algorithm and it returns an oversized bounding box. In motion blurred images the text disappears but the high corner density in the code area makes it still clearly distinguishable. Input images are from Dubská et al.

boxes, we achieve an overall detection ratio of only 42.6%.

On our QR data set with real sharp, defocused and motion blurred images (examples are shown in Figure 10) our algorithm can reliably detect 81% of the codes. However, if a code is rotated close to 45 °, our simple axis-aligned bounding box detection tends to return a bounding box that is too small.

4.3.3 Multi Code Performance

All described methods focus on localizing only one code in the image but they all build a barcodeness map in an intermediate step. Therefore, they all could be extended to localize multiple codes and provide a list of ranked candidates. Figure 7 shows two examples with multiple 1D and 2D codes together with our barcodeness maps. If both code types are present in the image the linear combination of m_1 and m_2 causes crosstalk in s_1 and s_2 which may lead to false classifications. Note that the codes are still visible but the code type might be uncertain. We leave the detailed analysis of this question for future work.

4.4 Discussion

In our 1D tests we achieve similar sharp detection rates as the others but without their limiting assumptions, while the algorithm also detects close to 17% of the blurry codes. This means in 17% of the cases a blurry decoder can already start decoding the barcode even before the AF was triggered which leads to better user experience.

The average runtime of our algorithm (on images of resolution 720x960) is 73ms that adds up as follows: HSV conversion (19.90 ms); gradient calculation (9.19 ms); calculation of C_{ij} (14.32 ms); calculation of saliency (6.19 ms); box filtering (7.51 ms); bounding box detection/rectangle detection (15.87 ms). The test PC has an Intel Core i7 M620 CPU with 2.67GHz clock frequency and the phone measurements were taken on a Samsung Galaxy Nexus. We can clearly see that the most time is spent on color space conversion. Without the HSV information the algorithm still finds sharp codes but it becomes more sensitive to clutter. Thanks to the saturation information our algorithm returns significantly less false positives than the others in our tests. It is also important to note that current smartphone models are also equipped with a graphical processing unit, and not

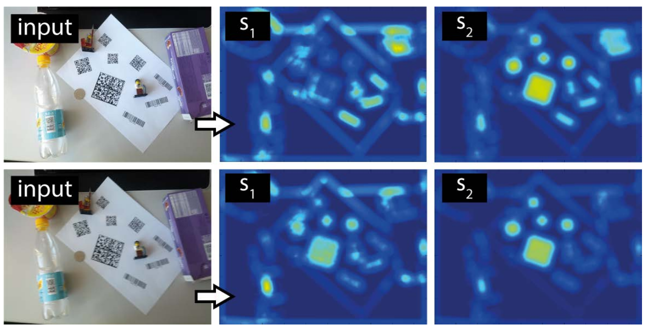

Figure 7: Multiple 1D and 2D codes in a sharp image (top row) and in a blurry image (bottom row). Note that the measures s_1 and s_2 clearly show the barcode areas in both the sharp and blurry images.

Algorithm	Gallo	Tekin	Katona	Ours
Runtime PC	27ms	49ms	63ms	73ms
Runtime phone	85ms	26ms†	173ms	380/118ms‡

Table 2: Average runtime of the four algorithms on PC (960x720 pixels) and on a smartphone (640x480 pixels). † Tekin is optimized C++ code, the other algorithms are written in OpenCV. Tekin is slower in our PC simulations because of image format conversions. ‡CPU only / calculation of s_1 and s_2 on GPU and reading back to CPU

just color conversion but every step of our algorithm can be implemented as OpenGL ES fragment shaders. We achieved a speedup of factor 3.2 by calculating the barcodeness maps on the mobile GPU.

Recall that for comparison we applied Gallo's method to generate a bounding box from the saliency map, a method that assumes axis-aligned barcodes in the image. While also rotated barcodes are well visible in our saliency maps, the final bounding boxes are sometimes too small ($J < 0.5$). A more sophisticated bounding box algorithm should significantly improve our detection rates.

While our algorithm is fully orientation invariant, it is not fully scale invariant. The size of the box filter is chosen to connect barcode areas of a fairly wide scale range, but it would not work with tiny codes (would include clutter) or huge codes (would break apart).

The resolution of current generation smartphone cameras is already high enough for decoding multiple codes that are cut out from the original image. The true advantage of scanning multiple small codes, however, will be unleashed with the upcoming smart glasses. Scanning multiple codes on a shelf may soon become possible without the need for holding the products close to the camera. Scanning several symbologies at the same time is important for enterprise applications. It is an interesting research question how to design a wearable user interface for barcode selection when multiple code candidates are available.

5. CONCLUSION AND FUTURE WORK

We have presented a combined 1D and 2D barcode localization algorithm that addresses orientation, scale, and symbology invariance and is also more robust to blur than previous methods. We compared the speed and accuracy of our approach to three recent localization algorithms on publicly available datasets and achieved higher detection ratio even with defocused images where previous methods perform weakly. Taking our algorithm as a preprocessing step, blurry barcode scanning can be extended to full-resolution images in the future. We are currently working on extending the bounding box detection to multiple codes in an image.

6. REFERENCES

[1] R. Adelmann. Mobile phone based interaction with everyday products - on the go. In *Proceedings of International Conference on Next Generation Mobile Applications, Services and Technologies*, NGMAST'07.

[2] J. Alfthan. Robust detection of two-dimensional barcodes in blurry images. Master's thesis, KTH Stockholm, Sweden, 2008.

[3] S. Ando. Image field categorization and edge/corner detection from gradient covariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(2):179–190, Feb. 2000.

[4] S. Ando and H. Hontani. Automatic visual searching and reading of barcodes in 3-d scene. In *Proc. IEEE International Vehicle Electronics Conference*, IVEC'01, pages 49–54, 2001.

[5] M. Dubská, A. Herout, and J. Havel. Real-time precise detection of regular grids and matrix codes. *Journal of Real-Time Image Processing*, pages 1–8, 2013.

[6] O. Gallo and R. Manduchi. Reading 1D barcodes with mobile phones using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1834–1843, 2011.

[7] D. Han, J. Teng, Z. Yang, Y. Pang, and M. Wang. 2D barcode image binarization based on wavelet analysis and Otsu's method. In *Proc. International Conference on Computer Application and System Modeling*, volume 5 of *ICCASM'10*, pages V5–30–V5–33, 2010.

[8] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. Fourth Alvey Vision Conference*, pages 147–151, 1988.

[9] M. Katona and L. G. Nyúl. Efficient 1d and 2d barcode detection using mathematical morphology. In *Proc. International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, ISMM'13, pages 464–475, 2013.

[10] H. W. Kongqiao Wang, Yanming Zou. 1D barcode reading on camera phones. *International Journal of Image and Graphics*, 7(3):529–550, 2007.

[11] A. Kutiyanawala, X. Qi, and J. Tian. A simple and efficient approach to barcode localization. In *Proc. International Conference on Information, Communications and Signal Processing, 2009*, ICICS'09, pages 1–5, 2009.

[12] E. Tekin and J. Coughlan. BLaDE: Barcode localization and decoding engine. Technical Report 2012-RERC.01, The Smith-Kettlewell Eye Research Institute, December 2012.

[13] S. Wachenfeld, S. Terlunen, and X. Jiang. Mobile multimedia processing. chapter Robust 1D barcode recognition on camera phones and mobile product information display, pages 53–69. Springer-Verlag, Berlin, Heidelberg, 2010.

[14] M. Wang, L.-N. Li, and Z.-X. Yang. Gabor filtering-based scale and rotation invariance feature for 2D barcode region detection. In *Proc. International Conference on Computer Application and System Modeling*, volume 5 of *ICCASM'10*, pages 34–37, 2010.

[15] W. Xu and S. McCloskey. 2D Barcode localization and motion deblurring using a flutter shutter camera. In *Proc. IEEE Workshop on Applications of Computer Vision*, WACV'11, pages 159–165, 2011.

[16] S. Yahyanejad and J. Ström. Removing motion blur from barcode images. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW'10, pages 41–46, 2010.

APPENDIX

A. EDGE AND CORNER MEASURES

To make our paper self-contained we present here the derivation of the applied edge and corner measures. We combine the derivations of Harris [8] and Ando [3] who both defined edge and corner measures based on the structure matrix M:

$$M = \begin{bmatrix} < I_x^2 > & < I_x I_y > \\ < I_x I_y > & < I_y^2 > \end{bmatrix} = \begin{bmatrix} C_{xx} & C_{xy} \\ C_{xy} & C_{yy} \end{bmatrix}$$

where I_x and I_y are the image derivatives in x and y direction. The entries C_{ij} are calculated over the D neighborhood of the pixel with a window function w:

$$C_{i,j} = \sum_{(x,y)\in D} w(x,y) I_i(x,y) I_j(x,y)$$

The window function is usually either a box window or a Gaussian window. The λ eigenvalues of this matrix are the solutions of the quadratic equation

$$\begin{vmatrix} C_{xx} - \lambda & C_{xy} \\ C_{xy} & C_{yy} - \lambda \end{vmatrix} = (C_{xx} - \lambda)(C_{yy} - \lambda) - C_{xy}^2 =$$

$$= \lambda^2 - (C_{xx} + C_{yy})\lambda + C_{xx}C_{yy} - C_{xy}^2 = 0.$$

Using Vieta's formulas, the two solutions λ_1 and λ_2 satisfy

$$\lambda_1 + \lambda_2 = C_{xx} + C_{yy} > 0, \quad \lambda_1 \lambda_2 = C_{xx}C_{yy} - C_{xy}^2 > 0$$

The discriminant of the quadratic equation is always non-negative:

$$(C_{xx} + C_{yy})^2 - 4(C_{xx}C_{yy} - C_{xy}^2) = (C_{xx} - C_{yy})^2 + 4C_{xy}^2 \geq 0.$$

This means that both λ_1 and λ_2 are real and non-negative. Actually, the two eigenvalues are the variances of the two principal components of the (I_x,I_y) distribution. By taking the ratio of their multiplicative average to their additive average, one can define the homogeneity measure m_2:

$$m_2 = \left(\frac{\sqrt{\lambda_1\lambda_2}}{(\lambda_1 + \lambda_2)/2}\right)^2 = \frac{4(C_{xx}C_{yy} - C_{xy}^2)}{(C_{xx} + C_{yy})^2}$$

Since if $\lambda_1, \lambda_2 > 0$ we have

$$0 \leq \sqrt{\lambda_1\lambda_2} \leq (\lambda_1 + \lambda_2)/2,$$

the measure m_2 is dimensionless and is normalized such that $0 \leq m_2 \leq 1$. Let us define a complementary measure m_1 as

$$m_1 = 1 - m_2 = \frac{(C_{xx} - C_{yy})^2 + 4C_{xy}^2}{(C_{xx} + C_{yy})^2}$$

which is also dimensionless and is normalized such that $0 \leq m_1 \leq 1$. To avoid division by zero in flat image regions, we add a small number ϵ to the denominators. Ando proves the following properties: (1) m_1 reaches 1 where the image intensity varies one-dimensionally (edges and ridges); (2) m_2 reaches 1 where the image intensity changes with circular symmetry or with rotational periodicity with a period $\pi/2$ (checkerboard corners). For more details please refer to [8] and [3].

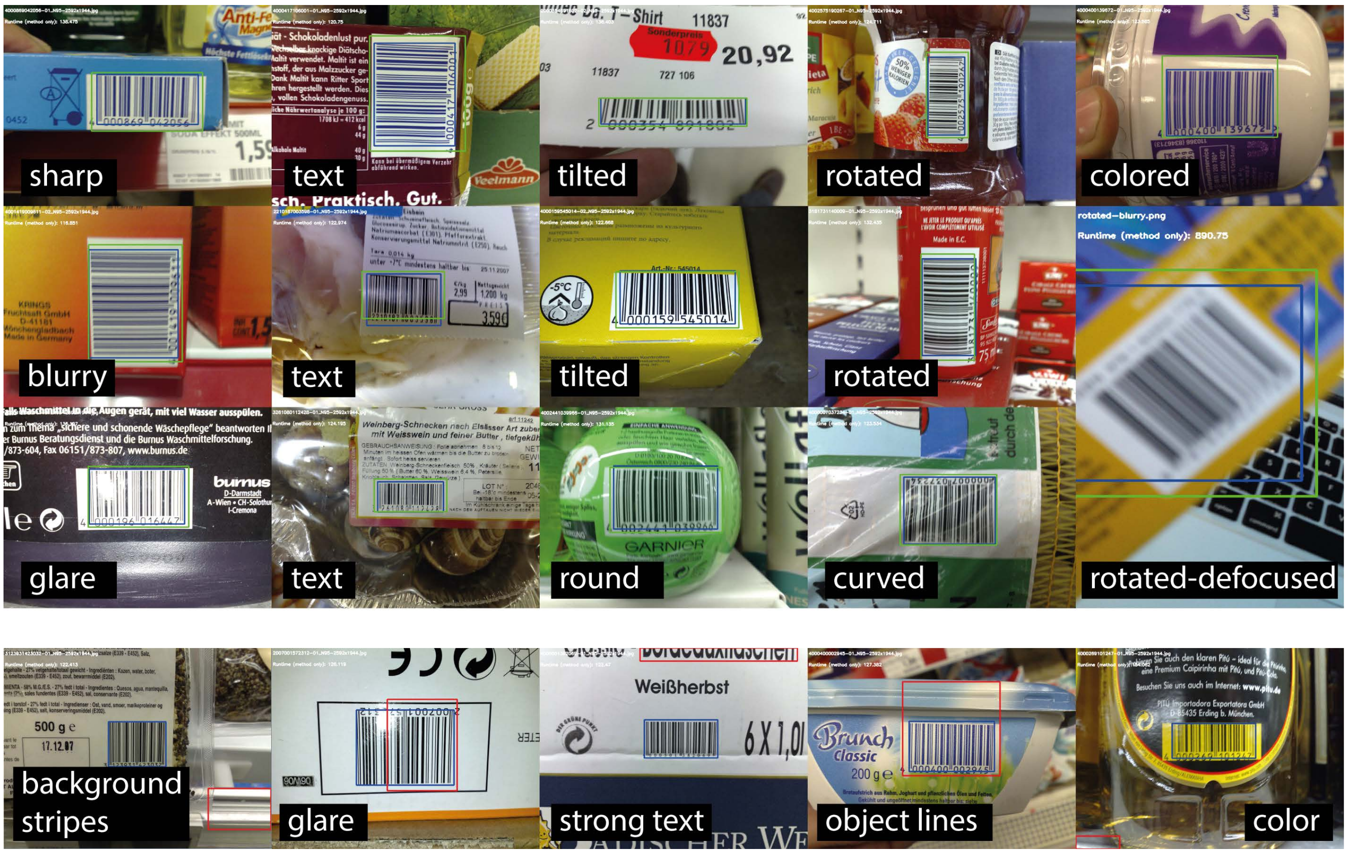

Figure 8: Positive and negative examples from the Muenster dataset. A green rectangle indicates positive detection ($J \geq 0.5$), a red rectangle indicates false detection ($J < 0.5$), blue rectangles represent the hand-clicked ground truth. Bottom row: The failures are caused by (1) parallel lines in the background, (2) glare, (3) dominant text, (4) object lines (5), colored code. (The runtimes given in the images also include loading files and opening windows)

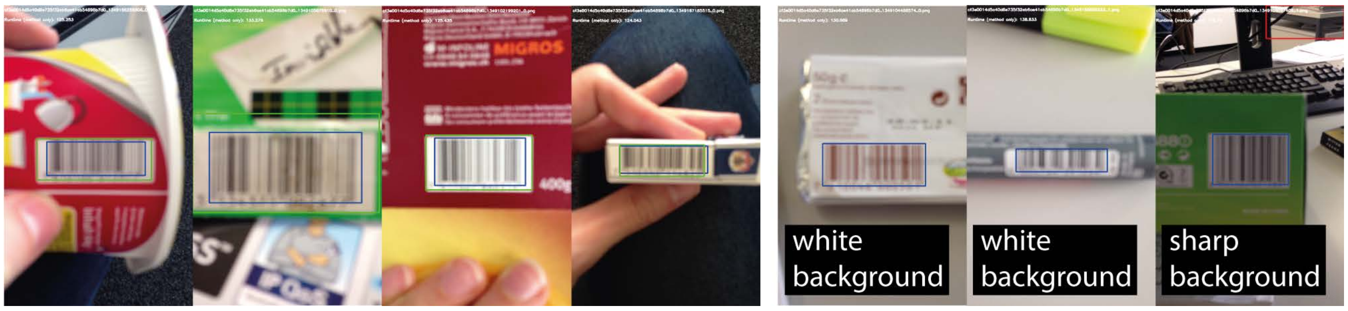

Figure 9: Positive and negative examples from our no-AF dataset. Left: Our algorithm can localize blurry codes with rectangular white support. Right: difficult cases with all-gray background and sharp background (no codes found).

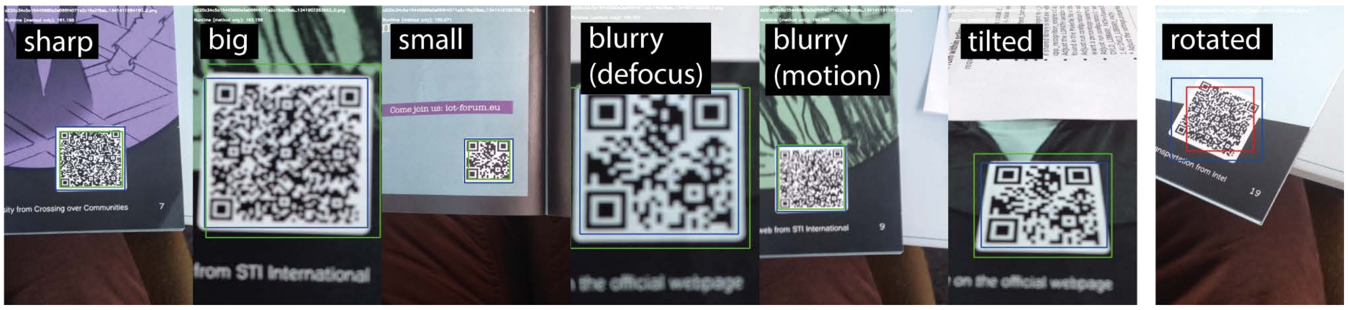

Figure 10: Positive and negative examples from our QR code dataset. Note that the misalignment on the right is caused by the simple axis-aligned bounding box detection step. The code area would be still clearly distinguishable in the s_2 map.