Fast Blur Removal for Wearable QR Code Scanners

Gábor Sörös, Stephan Semmler, Luc Humair, Otmar Hilliges
Department of Computer Science
ETH Zurich
{gabor.soros|otmar.hilliges}@inf.ethz.ch, {semmlers|humairl}@student.ethz.ch

Figure 1: Left three: Scanning visual codes with handheld and wearable devices is becoming ubiquitous. Right four: When the camera or the scanned object move even slightly during scanning, the captured visual codes become distorted by motion blur and conventional decoder algorithms cannot decode them. We propose a fast blur removal algorithm that allows scanning motion-blurred QR codes.

ABSTRACT

We present a fast restoration-recognition algorithm for scanning motion-blurred QR codes on handheld and wearable devices. We blindly estimate the blur from the salient edges of the code in an iterative optimization scheme, alternating between image sharpening, blur estimation, and decoding. The restored image is constrained to exploit the properties of QR codes which ensures fast convergence. The checksum of the code allows early termination when the code is first readable and precludes false positive detections. General blur removal algorithms perform poorly in restoring visual codes and are slow even on high-performance PCs. The proposed algorithm achieves good reconstruction quality on QR codes and outperforms existing methods in terms of speed. We present PC and Android implementations of a complete QR scanner and evaluate the algorithm on synthetic and real test images. Our work indicates a promising step towards enterprise-grade scan performance with wearable devices.

Author Keywords

QR code; visual code; motion blur; deblurring; mobile

ACM Classification Keywords

I.4.4 Image Processing and Computer Vision: Restoration; H.4.2 Information Systems Applications: Logistics

INTRODUCTION

Visual codes printed on physical objects of all kinds play an important role in many consumer and research scenarios that aim to embed digital information in the real world. Quick response (QR) codes are particularly popular and are found in numerous applications, such as scanning codes for shopping or product comparison, checking electronic tickets, delivering parcels, and are found in magazines, on postcards and other adverts (see Figure 1, left). Finally, QR codes are also becoming ubiquitous in enterprise asset tracking and provide employees access to detailed object records.

On most current mobile platforms such as smartglasses and smartwatches, QR code scanners are available for free. Current scanning solutions work well only if the camera is held still and relatively close to the scanned object (Figure 1, 2^{nd} left). However, with wearable cameras, slight motions during scanning are almost unavoidable and motion can easily render the captured codes unreadable (Figure 1, right).

Under motion blur, conventional decoder algorithms fail to recognize even slightly unsharp codes which greatly deteriorates user experience and utility. Scanning out-of-focus or very small codes is another challenge that camera-based wearable scanners must overcome in order to become competitive with commercial-grade laser scanners. The common aspect in these related image degradations is an underlying mathematical blur model that, when carefully inverted, allows for removing blur from the images. We present and algorithm that can robustly decode QR codes degraded by synthetic or real motion blur. Furthermore, we show promising results with synthetic defocus blur and upscaling blur.

Problem statement and contributions

The problem of removing blur from photographs has been widely studied in the past but existing algorithms typically fail on artificial black and white visual tags because they look very different from natural images. We illustrate this in Figure 5 where we compare our QR restoration approach with other deblurring methods from the literature. Intuitively, text deblurring appears to be a similar problem to ours. However, text deblurring methods expect a few thin black lines on dominant white background. Visual codes, in contrast, usually have an equal distribution of dark and light areas. Furthermore, the existing restoration algorithms are computationally

ISWC '15, September 7–11, 2015, Osaka, Japan.

http://dx.doi.org/10.1145/2802083.2808390

too demanding to be carried out on mobile devices – even if they make strong assumptions about the image type. We are not aware of any published method that achieves anywhere near real time blur removal performance (even on PCs). However, the special structure of visual codes compared to general photographs allows for optimizations in terms of restoration speed and, to a lesser extent, quality. We can exploit two important differences between restoring general photographs and restoring QR codes. First, QR codes do not need to *look* perfectly for decoding. Second, the checksum[1] in the codes allows for early termination while it also eliminates false positive detections.

Our main contribution is a new practical *blur removal algorithm* specifically tailored for decoding motion-blurred QR codes on CPU- and memory-constrained wearable computers. We improve general purpose deblurring methods from the literature by adapting each step to the *specific properties* of QR codes that ensures fast convergence to the correct solution. We also propose and empirically evaluate a *new initialization scheme* that greatly improves convergence and the quality of the results in removing large motion blur. We present *fast PC and Android implementations* and show in *thorough experiments* that our iterative restoration-recognition algorithm can quickly decode QR code images degraded by *synthetic or real motion blur*. From the *comparison with the state of the art* we conclude that our restoration *quality is on par* with existing methods while the restoration speed is about a *magnitude faster*.

RELATED WORK

Removing motion blur from an image is a mathematically challenging inverse problem. A common assumption [2, 3, 6, 10, 11, 15, 13, 19] is uniform (shift-invariant) blur over the image which simplifies the mathematical models and allows for faster restoration algorithms. The uniform blur process can be described as a convolution of the sharp image with a blur kernel hence blur removal is also termed deconvolution. Non-blind deconvolution refers to deconvolution with a known kernel. In contrast, in blurry QR scanning the kernel needs to be estimated first, this is called blind deconvolution.

Existing blind deconvolution algorithms usually follow a common pattern of Bayesian energy minimization. The unknown latent sharp image and blur kernel are estimated successively in a multi-scale iterative optimization scheme. Usually, natural image statistics (given distribution of image gradients) are applied as additional constraints on the restored image [6, 7]. The blur kernel can be estimated for instance from edge profiles [5, 3, 4, 17], from external sensors [12], or using special camera hardware [21]. In this paper, we target unmodified wearable devices, so our attention is on the family of edge-based methods, and only the fastest of those. These algorithms in a first step try to hallucinate sharp edges, and in a second step find the blur kernel that causes the observed blurry edges. The two steps are alternated in an iterative optimization, often also across multiple scales to aid the convergence.

[1]We refer to the QR error detection and correction capability.

Deblurring natural images

Cho and Lee [3] presented the first fast algorithm for motion deblurring. The main idea is that for image reconstruction a low-quality but fast step is sufficient if the errors are suppressed and edges are boosted by edge-aware image filters. They achieve quality comparable to previous attempts within a few seconds, up to two magnitudes faster than others. Xu and Jia [19] analyzed which edges are actually useful for kernel estimation and showed that structures smaller than the kernel size (like thin lines of a barcode or modules of a QR code) may mislead the optimization. They proposed a filtering approach that selects strong edges in the image. Other methods like [4] and [5] recover the blur kernel by explicitly inspecting how sharp edges of the scene get blurred and reconstruct the kernel from its cross-sections. In [20] the authors exchange the parametric edge enhancement filters of [3] with a new regularization term that approximates the L_0 norm of the gradients. With this new energy formulation, the algorithm no longer relies on ad-hoc edge selection and filtering methods which means no parameter tuning is required. [13] revisits an older method called total variation deconvolution, and analyzes in details why – although it is simpler than natural image priors – can still find the correct sharp solution.

Overall, the general photograph deblurring algorithms today have high computational requirements and are tuned for natural scenes. They often fail on artificial image content such as visual codes because of the different appearance.

Deblurring text and visual codes

Previous work also addressed defocus and motion blur removal from text and barcode images. While the shape of a defocus kernel is given by the lens characteristics and can be measured accurately, a motion blur kernel can have very different shapes depending on the object or camera motion, hence compensating the latter is significantly more difficult.

Compensating defocus blur in 1D barcodes can be considered solved and fast algorithms exist in commercial barcode scanner applications[2]. Liu et al. [8] extended a standard deconvolution method with a bi-level image histogram constraint for quickly removing defocus blur from 2D DataMatrix codes, but this method would not work with motion blur.

Compensating motion blur in 1D barcodes is addressed in Yahyanejad et al. [22]. They reduce the problem to 1D by averaging over several lines of the barcode, which renders the method inapplicable for 2D QR codes. Xu and McCloskey [21] presented a motion deblurring method using a camera with a fluttered shutter, a hardware modification that makes the blur easier to invert. We target off-the-shelf wearable devices without such modifications. Gennip et al. [17] presented an algorithm for blind deblurring of QR codes by explicitly making use of the known finder patterns to estimate the kernel. The algorithm is evaluated only with synthetic blurs, and with the assumption that the location of the blurry finder pattern is known, which might be difficult to determine in real images.

[2]e.g., RedLaser **www.redlaser.com**, Scandit **www.scandit.com**

Our QR restoration algorithm is closely related to recent research on blind text deblurring. The method of Cho et al. [2] can successfully remove motion blur from text images that have a lot in common with visual tags, but the algorithm relies on precise text segmentation and is too slow for our purposes. The text deblurring method of Chen et al. [1] is specifically developed for binary text images and is unlikely to work well with cluttered images. The algorithm of Pan et al. [10] applies L_0-minimization not only to image gradients (see [20] for natural images) but also to the pixel values which means the algorithm enforces the image to have a few black pixels among many white pixels, which is typical for documents. Our experiments have shown that this method only works for QR codes if its parameters are carefully tuned. Furthermore, the algorithm is too slow for mobile applications.

FAST QR DEBLURRING

Instead of high-quality deblurring, we rather focus on high-speed decoding. We contribute an algorithm that is particularly suitable for fast restoration of a single QR image.

Properties of QR deblurring

In addition to the findings made in the general blind deconvolution techniques, we can make the following observations. (i) QR codes contain many black and white corners that are easy to localize even in a blurred image. (ii) QR codes include a checksum, so the algorithm can terminate when the checksum is correct. False positives are hence practically impossible. (iii) QR codes contain strong error correction, so even partially restored codes may be decoded. This is especially important when the blur is slightly non-uniform in the code area. (iv) QR codes consist of sharp edges, which is advantageous for blur estimation, but disadvantageous for blur removal. (v) QR codes also contain small structures that may mislead the blur estimation process [19] and therefore their influence needs to be suppressed.

Method overview

We apply the uniform blur model which describes the blurred image B as a convolution of a latent sharp image I and a blur kernel k, with additive Gaussian noise N:

$$B = k * I + N \tag{1}$$

Uniform blur is a valid assumption if the blur is caused mainly by translational motion. Cropping the image to a small search region in practice reduces blur to mostly this kind.

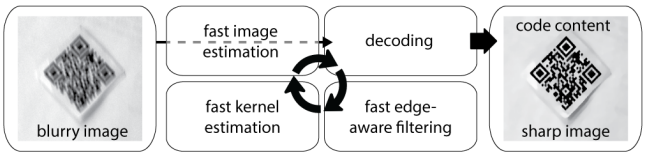


Figure 2: Method overview

The algorithm consists of four main components that work in a tightly coupled restoration-recognition loop (see Figure 2). In the first step, we run a standard *decoder* algorithm; if an image is sharp enough for decoding, the loop terminates. Otherwise, the contrast is increased in the second step and *edge-aware filtering* is applied. This removes noise from the image, flattens small structures while it keeps strong edges. Next, the blur *kernel estimation* step finds the most likely kernel which transforms the sharpened image into the blurry one. In the fourth step the blurry image is deconvolved with the currently estimated kernel to get a sharper *image estimate*, which again enters the decoder. The process is iterated up to five times to attain improved kernel estimates and sharper image estimates. Performing the loop over multiple image scales ensures fast convergence to the correct solution.

Similar to other blur removal techniques, we formulate the QR deblurring problem as an energy minimization scheme over I and k and iteratively optimize for one while keeping the other constant. The total energy function to be minimized consists of a data fitting term and regularization terms $\rho_I(I)$ on the image and $\rho_k(k)$ on the kernel with regularization weights λ and γ, respectively. For the data fitting term, the L_2 norm is commonly used, and different blind deconvolution methods apply different regularization terms. The common form of the energy function is

$$\arg\min_{I,k} ||B - k * I||_2^2 + \lambda\rho_I(I) + \gamma\rho_k(k) \tag{2}$$

Our goal is to find a good compromise between restoration quality and restoration speed, also taking into account the typical black and white structure of QR codes. We chose a sparsity prior on the image gradients ∇I_i and an L_2 sparsity prior on the kernel values k_i:

$$\rho_I(I) = \sum_{\forall i} |\nabla I_i|^\alpha \qquad \rho_k(k) = ||k||_2^2 = \sum_{\forall j} k_j^2 \tag{3}$$

In our notation, i and j index image pixels and kernel pixels, respectively. The optimization of the total energy function can be separated into I- and k-subproblems that are presented in the following sections.

Fast image estimation

Assuming a current estimated kernel k is known, we apply the fast *non-blind* deconvolution method of Krishnan [6] that in general enforces a sparse hyper-Laplacian distribution on the gradients of the sharp image. With the exponent $\alpha = 1$, the enforced Laplacian distribution of the gradients in turn corresponds to a total variation regularization of the image I. This is well suited for QR-codes as it favors flat image regions while it also allows sharp edges.

$$\arg\min_{I} ||B - k * I||_2^2 + \lambda||\nabla I||^\alpha \tag{4}$$

For $\alpha = 1$, the solution is particularly simple and fast, relying only on FFTs and thresholding operations [6].

To reduce boundary artifacts in the FFT-based restoration, we wrap the image boundaries using the method of Liu and Jia [9] prior to deconvolution. This wrapping method is suitable for not only symmetric but general kernels, is reasonably fast, and can be used in any FFT-based restoration method.

Fast edge-aware filtering

Due to an imperfect kernel, the fast deconvolution produces unwanted ringing artifacts and noise that need to be suppressed while the main structure of the image must be kept

unchanged before kernel estimation. At this stage other deblurring methods usually apply a combination of bilateral and shock filters [3] which require parameter tuning, or L_0-smoothing [20, 10] which is slower. We propose to use the joint weighted median filter (WMF) [23] to remove small variations in the structure. This new filter produces an output similar to other edge-aware smoothing filters but is significantly faster. The filter radius is set $1/5$ proportional with the current kernel size. The blurry B and the sharpened I image pair is passed to the next stage for blur kernel estimation.

Fast kernel estimation

Given an image estimate I, we solve for the kernel k in the gradient space using the efficient method of Cho and Lee [3]. The energy function to be minimized here is

$$\arg\min_k ||\nabla B - k * \nabla I||_2^2 + \gamma ||k||_2^2 \qquad (5)$$

which can be solved with the conjugate gradient method [3] in the Fourier domain where convolution turns into multiplication. The L_2-norm on k favors sparse solutions which is desirable because a motion blur kernel consists of a thin continuous motion path. Working with image gradients instead of image intensities is crucial here because it allows to ignore boundary artifacts which drastically reduces the number of FFTs required [3].

Next, potentially disconnected small components in the kernel are discarded and the kernel is shifted to its geometrical center. Other methods usually shift the kernel to its mass center but that might clip long thin tails at the boundaries. Finally, the kernel is normalized so that the convolution does not reduce or increase the energy of the image.

Decoding in a restoration-recognition loop

After each iteration, we let a common QR detector and decoder algorithm process the image. The error correction in QR codes practically disables false decoding while it can guarantee that the algorithm converged to the right solution.

We perform our calculations over multiple image scales, starting with a kernel size of 5×5 up to 33×33 pixels. The image pyramid is built with scale factor $1/\sqrt{2}$. On each scale level the algorithm performs up to 5 iterations. Between scale levels, the kernel is bilinearly upscaled. Figure 3 illustrates how the kernel gets refined over the iterations.

Once the QR code is recognized, depending on its content a URL is opened or the contact details are shown to the user.

Initialization

One of our main contributions is a grid-shaped starting kernel. Other methods usually initialize the kernel either with a Dirac delta function (identity blur) or with a small 2D Gaussian (small defocus blur). In contrast, we initialize the kernel with a grid of Dirac functions (see Figure 3) which corresponds to multiple shifted copies of the sharp image after convolution. The proposed kernel has, to our knowledge, not been reported in the literature but makes an important difference, in particular for heavily blurred images. While the delta kernel works well in simulations, we found in experiments

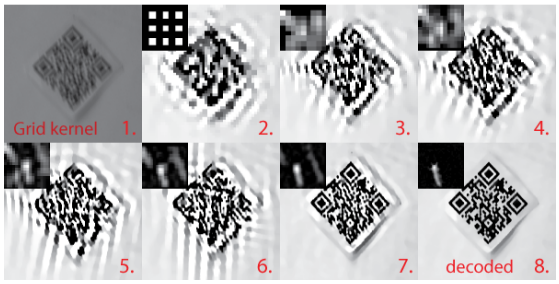


Figure 3: Illustration of image and kernel refinement over 8 iterations using a grid of peaks as starting kernel. The example also illustrates how disconnected kernel noise gets removed during the process. The image is 300×300, the kernel is 33×33 pixels, the decoding took 3.107 seconds. The blurry image was taken with a smartphone.

with real smartphone images that using a grid as initial kernel greatly improves the robustness of the algorithm. We provide an analysis on the impact of the new kernel in the evaluation section. However, this gain in deblurring performance comes at a small cost as in general the grid kernel converges slower than the delta kernel. Fortunately, one could overcome this tradeoff by reading the inertial sensors during image capture and thus selecting the right initial kernel based on a first estimate of the blur size. Due to constraints of the Android API, we have left this for future work.

Implementation

We have implemented the algorithm on both PC and Android using the open-source OpenCV, FFTW, and ZBar libraries[3]. We have found the constants $\lambda = 0.002$, $\gamma = 2$, and WMF standard deviation 55 to work well in all our experiments. The algorithm requires no further parameter tuning. Our scanner is built as a standard native Android application which makes it portable to any Android device.

Our linear blur model does not take into account the manufacturer-dependent non-linear image enhancements and other effects in the camera that may cause significant impact on the performance of blur removal algorithms [16]. It is therefore vital that we set the tone map curve to linear[4] and switch off the automatic image enhancement functions in our camera. In our experiments, this is possible through the Android *Camera2* API. This fine camera control is available since Android 5.0 if the camera driver also supports it. This fact, unfortunately, limits our live tests to high-end smartphones, however, we expect that with rapid technological advancement the smartphones' capabilities will soon be available in smartwatches and smartglasses as well. Until that, the smartglasses implementation can be tested using synthetic images, which is sufficient for speed and memory analysis. The graphical user interface of our smartphone application is also shown in Figure 8, and the smartglasses interface is shown in the accompanying video.

EVALUATION

We evaluate the effectiveness of our algorithm on a series of QR code images contaminated by synthetic and real blur,

[3] `www.opencv.org`, `www.fftw.org`, `www.github.com/ZBar`

[4] This is the reason why our input images appear dark

and also compare our algorithm with the state of the art. For convenience, the comparisons were performed on a notebook with a 2.40 GHz Core i7-4700MQ CPU. Our algorithm currently uses a single thread only. The input images with synthetic blur were created in Matlab, the input images with real blur were captured with a Google Nexus 6 smartphone. We also show qualitative results of our algorithm running directly on the smartphone and on smartglasses.

Removing synthetic motion blur

In the first experiment, we test whether the algorithm can remove synthetic uniform blur, i.e., blur that our mathematical model assumes. Figure 4 illustrates the experiment. For repeatability, we use 6 out of the 8 benchmark kernels from Levin et al. [7]. We leave out 2 kernels that are so small that we can decode the images without deblurring. The input images are of resolution 200×200, the kernels are between 17×17 and 27×27 pixels. In this experiment, we use small images because the benchmark kernels are rather small. To simulate imperfections of real images, we also add small 0.1% noise to the blurred images. The top row of Figure 4 shows the input images and in the insets the ground-truth blur kernels that were used to create the input images. The bottom row shows each intermediate image and the current estimate of the kernel when the code first could be decoded. The output images and kernels are upscaled for visualization using nearest neighbor interpolation. The form of our estimated kernels resembles that of the ground truth kernels. On the PC, it takes about 0.450 s to restore the first five images while the last one takes 1.6 s because iterations on two more scales are required due to larger blur.

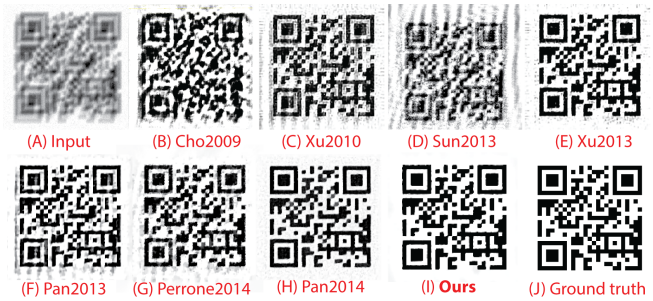
Figure 4: Removing synthetic uniform blurs. Top: input images and ground truth kernels. Bottom: output images and kernel estimates when first decoded. Kernel size indicates the scale level.

We repeated the experiment after adding large 1% image noise. The algorithm can successfully decode all blurry and noisy examples with the expense of a longer average runtime of 0.657 s. The images can be found in the supplement.

We conclude that our algorithm is able to quickly restore blurry codes that follow our mathematical model, however, the restoration speed depends on both the image size and blur size. The main bottleneck of real-time operation is the number of single-threaded FFTs but we expect a speedup with a parallel implementation.

Comparison of blind deblurring methods

For practical applications, the restoration speed is crucial. In this experiment, we compare our algorithm with the state of the art in terms of runtime and readability of the resulting images. All selected methods are available open source or as executables. Methods *(B)-(G)* were developed for general blind deconvolution problems while method *(H)* is specifically for text image deblurring. A short summary of each method can be found in the Related work section. Figure 5 summarizes our findings.



	A Input	B Cho2009	C Xu2010	D Sun2013	E Xu2013
binary	-	C++/GPU	C++/GPU	Matlab	C++/GPU
runtime	-	0.481s	0.955s	217.730s	1.049s
decoded	-	no	no	no	**yes**
	F Pan2013	G Perrone2014	H Pan2014	I Ours2015	J Truth
binary	Matlab	Matlab	Matlab (*C++*)	C++	-
runtime	133.8s	171.898s	12.736s (*9.691s*)	1.765s (**0.614s**)	-
decoded	no	**yes**	**yes**	**yes**	-

Figure 5: Comparison of blind deconvolution algorithms on a synthetically blurred QR code. (A) Input frame (B) Cho2009 [3] (C) Xu2010 [19] (D) Sun2013 [15] (E) Xu2013 [20] (F) Pan2013 [11] (G) Perrone2014 [13] (H) Pan2014 [10] original Matlab implementation and in brackets *our C++ port* (I) Ours, in brackets the time of first decoding (J) Ground truth

Method (B) [3] was the first fast algorithm for natural image deblurring, but the applied gradient statistics make it fail on artificial black and white content. Also, this method is sensitive to initialization parameters. Method (C) [19] selects good edges for kernel estimation and reconstructs the sharp edges but the result exhibits significant ringing artifacts that make the code unreadable. Method (D) [15] decomposes the image into a dictionary of sharp edge and corner patches to aid the kernel estimation. The method is very slow, and although the patch prior was expected to fit well with the structure of QR codes, the result is surprisingly unreadable. Method (E) [20] is fast and can successfully restore the code. However, to get good results, we had to supply an estimate for kernel size almost double of the true kernel size which is rather inefficient in terms of calculation overhead.

Method (F) [11] also implements an edge selection strategy, but the decoding failed. Method (G) [13] uses TV regularization like we do. It can successfully restore the code but is very slow. Method (H) is especially interesting as text images share similar properties with visual codes. We tested the authors' Matlab implementation and we also ported the algorithm to C++ to make it faster (we achieved about 30% speedup, not using the GPU at all). The result of both versions is almost perfect. The runtime was 12.736 s in Matlab and 9.691 s in our C++ implementation. The critical bottleneck is the L_0 restoration step. Also, the algorithm requires tuning many parameters, we found that QR codes need different regularization weights than text.

Our method (I) almost perfectly reconstructed the code (c.f. ground truth (J)). Further advantages of the proposed method over the reconstruction quality are its low memory footprint and high speed. Our method is almost as fast as (E), and it is important to note that (E) performs the FFT calculations on the GPU while we use one CPU core only. Also note that Matlab performs parallel processing in several built-in functions, so an exact runtime comparison is difficult.

Removing real motion blur

Next, we test the algorithm on codes contaminated by real motion blur. We capture a sequence of images with a smartphone while holding a QR code in front of the camera and deliberately shaking either the code or the smartphone. We used the same QR code with error correction level M in all our experiments. We capture camera preview frames because barcode scanner applications usually work with those instead of still images. We set the frame size to 720×480 pixels. We tested 340 images in total, out of this 217 frames were decoded without deblurring (63.8%), 83 frames were decoded after deblurring (24.4%), and only 40 frames were unsuccessful (11.8%), so our algorithm significantly increased the number of codes that could be scanned. Figure 6 shows examples of the restored codes. The decoding of these blurry examples took on average 1.4 s on the PC, but the number of iterations and so the speed depends on the size of the blur.

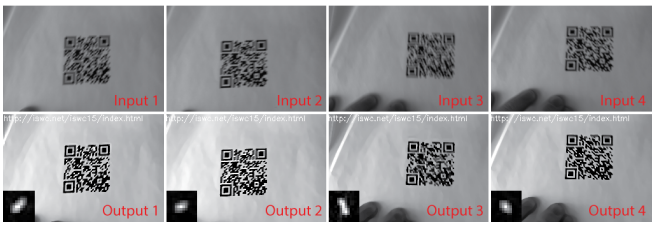


Figure 6: Removing real motion blur from QR code images. The decoded content (ISWC2015 URL) is written in the top of the images.

Impact of initial kernel choice

As discussed before, one of our main contributions is the proposed initialization scheme. In experiments with real data we have experienced that in case of heavy blur and when initialized with the Dirac kernel, the algorithm often did not converge to the right solution. In her seminal work, Levin [7] analyzed why blind deconvolution algorithms may converge to a blurry solution, however, the problem is different here, because we observed sharp but unreadable output images. We then introduced a new grid-shaped starting kernel that with large blurs works better than the Dirac kernel. In Figure 8 and in the supplemental material we show several example results of our experiments that verify our claims. We assume the better performance of the grid kernel is in connection with the regular structure of the underlying sharp QR code, but a rigorous analysis of this finding has not been done yet and a thorough explanation is left for future work.

Experiments on smartphones and smartglasses

After successful offline experiments with real motion-blurred images, we ported our algorithm to Android for live experiments. Figure 8 illustrates a concrete example with a large QR code and challenging blur restored directly on the Nexus 6 smartphone (Qualcomm Snapdragon 805 SoC with 2.7 GHz Krait 450 CPU and 3 GB RAM). The Android GUI consists of the camera preview and three image views for the input, output, and kernel images, respectively. The camera resolution is set to 720×480, and we added a 300×300 search window on top of the preview and constrain the algorithm to this area. Additionally, there are buttons to capture an image and start the deblurring, and a textbox for logging output. The challenging image is decoded in about 13 s.

We also implemented the algorithm on more constrained Google Glass smartglasses (TI OMAP 4430 SoC with 1.2 GHz CPU and 2 GB RAM). As the tonemap curve cannot be controlled, the recorded images on the Glass do not follow our linear blur model. Due to this limitation of the camera driver, we can report the performance on the Glass only using synthetically blurred images. Deblurring a 300×300 image on the Glass takes 8.54 s (using a single CPU core), which is about $2.5\times$ slower than deblurring the same image on the smartphone. The Android debugger reports the use of 34.9 MB of memory during deblurring, which we find acceptable on a wearable device. More experiments can be found in the supplementary video.

DISCUSSION

Target devices

As shown in the experiments, the algorithm requires a fast processor, moderate amount of memory (less than 40 MB without strict optimizations), and a camera with a suitable driver that allows manual control. As usually more complex applications are built on top of barcode scanning, the high CPU requirements restrict the applicability of our approach to high-end wearable devices. Slower wearable devices (e.g., a smartwatch or a life logger camera) could send the captured image to the user's smartphone which we believe is the most powerful wearable device nowadays. A cross-device barcode scanning solution (i.e., smartwatch camera, smartphone processor, smartglasses display) should be further explored in future research. However, distributing the computation on many individual devices would produce too much communication overhead. In the future, the algorithm could greatly benefit from embedded processors with fast floating point calculations and/or DSP support with hardware FFT features.

Camera resolution and code size

An important setting of the algorithm is the image size. We chose a 300×300 search window with the common 720×480 preview resolution. This corresponds to a convenient 15 cm scanning distance for a 5×5 cm code, a typical QR shopping scenario with the smartphone. The Glass camera has a wider field of view, so with the same resolution and search window, the code must be placed closer to the camera. In order to match the search window with a smaller code that the user holds further away, the camera resolution needs to be increased. Doubling the camera resolution means the code is visible from twice the distance, but then a twice higher resolution blur kernel ir required to represent the same shake blur. For example with preview resolution 1280×960 on the Glass, the 300×300 search window fits a 5×5 cm code in 35 cm

distance. As only the search window is processed, the actual camera resolution makes no difference in the image estimation, but the higher resolution makes a big difference in the blur kernel estimation.

Speed optimizations

The runtime of the algorithm depends on several factors. Figure 7 illustrates the time spent on image estimation and kernel estimation on different scales when the early detection is switched off. The complexity of image estimation grows exponentially with the image size, and the complexity of blur estimation grows exponentially with the blur kernel resolution. The complexity of FFT is known, the complexity of conjugate gradients can be estimated, but the number of iterations and scales until decoding depends on the actual blur shape. As the total time per scale grows exponentially, the importance of a fixed search window and early QR detection is indisputable.

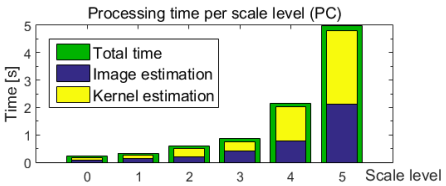


Figure 7: Time spent on image and kernel estimation on different scales (PC). In this example, the algorithm with QR detection would terminate successfully after 3 scales.

The runtime of the algorithm could be further reduced. We provided speed measurements of our single-threaded implementation, but as the algorithm contains a large number of FFTs, it could be greatly accelerated with parallel processing on multiple cores or even mobile GPUs or DSPs. Further speedup in wearable scanners could be achieved with inertial sensors that can aid the kernel estimation. The sensor stream captured together with the camera stream allows for reconstructing the camera motion during the exposure time which in turn relates to the (non-uniform) blur in an image. However, the lack of precise synchronization of the camera and the sensors on current wearable platforms is prohibitive without hardware modifications.

During scanning, we have access to multiple images of the QR code which could be exploited in multi-frame deblurring algorithms. However, those require precise image alignment which is difficult under arbitrary blur. In some cases, the camera might capture another sharp image during processing, so it might be advantageous to run a cheap decoder in parallel with deblurring. In this paper, we focused on scenarios when only a single blurry image of the code is available, typical in low lighting conditions, or when the code is moving.

Limitations

Our algorithm can estimate and remove uniform blur only, therefore it fails in case of significant rotational motion or strong rolling shutter distortions in the image (see video supplement). Blind non-uniform deblurring is computationally too demanding [18] to perform on current generation mobile devices. However, the uniform blur assumption is usually valid in the search window of our user interface.

We assume QR-specific gradient statistics in the whole restoration window which is violated if the code is placed in front of a complex background. To succeed, either the background must be a plain color, or the code must be segmented in a preprocessing step, for which algorithms do exist [14].

Future Work

So far, we have focused on decoding motion-blurred codes only, but the QR properties remain the same under other types for blur as well. In our future work we will investigate the adaptations required in kernel regularization to allow different non-sparse shapes. In the supplement, we briefly show our promising results in removing synthetic defocus blur and synthetic upscaling blur.

APPLICATIONS AND IMPACT

A blurry QR scanner brings many advantages for both consumer and enterprise applications.

For consumers, shopping becomes even easier by simply turning the product's code to the smartglasses while putting into the shopping basket. Robust blur removal from upscaled codes allows scanning tiny codes in the environment that is a core step in many applications of ubiquitous computing. Fast compensation of motion blur means conductors can have a ticket reader more robust to sudden movements in trains. We also imagine a new type of point of sale system with a simple tablet computer on the table. Products can then be checked out by simply swiping the code above the front camera.

In industrial applications, logistics employees and factory workers need to carry expensive enterprise handhelds to scan their own visual tags. While the proprietary protocols of these devices are rather difficult to integrate into business applications, they are also expensive and hence available to a limited number of employees only. On the other hand, each employee has a smartphone in the pocket (soon maybe smartglasses as well) with outstanding processing and sensing capabilities, with easy application development, and with an intuitive user interface. From a business process perspective, there lies great potential in a ubiquitous, smartphone or smartglasses scanning solution because then every employee can have a programmable barcode scanner and can access information on every item across the value chain. For the post and the packaging industry, blurry QR scanning saves time because codes can be scanned without stopping the conveyor belt.

The above examples illustrate that fast blur compensation could even create new use cases for barcode scanning that are not possible with today's technology.

CONCLUSION

We have presented a method for reading severely blurred QR codes in images casually captured on the go with mobile devices. Our restoration-recognition algorithm brings cheap wearable QR scanning one step closer to enterprise-grade performance and bears great potential for practical applicability. Our method can help to bring the comfort, the productivity, and the new business opportunities of wearable scanning also to the professional users who are accustomed to the performance of their laser scanners.

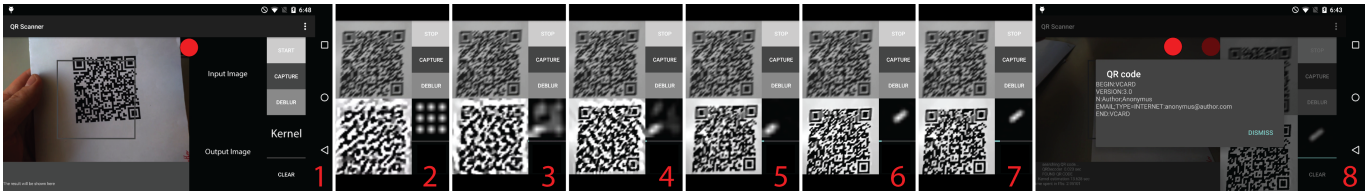


Figure 8: Android GUI: This example shows removing large motion blur from a QR code in 13.638s directly on a smartphone. The input is replicated in the top row and intermediate steps are shown in the bottom row. More examples with smartphones and smartglasses can be found in the supplementary video.

REFERENCES

1. Chen, X., He, X., Yang, J., and Wu, Q. An effective document image deblurring algorithm. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2011).
2. Cho, H., Wang, J., and Lee, S. Text image deblurring using text-specific properties. In *European Conference on Computer Vision (ECCV)* (2012).
3. Cho, S., and Lee, S. Fast motion deblurring. In *ACM SIGGRAPH Asia* (2009).
4. Cho, T. S., Paris, S., Horn, B., and Freeman, W. Blur kernel estimation using the radon transform. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2011).
5. Joshi, N., Szeliski, R., and Kriegman, D. PSF estimation using sharp edge prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2008).
6. Krishnan, D., and Fergus, R. Fast image deconvolution using hyper-Laplacian priors. In *Advances in Neural Information Processing Systems (NIPS)* (2009).
7. Levin, A., Weiss, Y., Durand, F., and Freeman, W. Understanding and evaluating blind deconvolution algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2009).
8. Liu, N., Zheng, X., Sun, H., and Tan, X. Two-dimensional bar code out-of-focus deblurring via the increment constrained least squares filter. *Pattern Recognition Letters 34*, 2 (2013).
9. Liu, R., and Jia, J. Reducing boundary artifacts in image deconvolution. In *IEEE International Conference on Image Processing (ICIP)* (2008).
10. Pan, J., Hu, Z., Su, Z., and Yang, M.-H. Deblurring text images via L0-regularized intensity and gradient prior. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014).
11. Pan, J., Liu, R., Su, Z., and Gu, X. Kernel estimation from salient structure for robust motion deblurring. *Signal Processing: Image Communication 28*, 9 (2013).
12. Park, S., and Levoy, M. Gyro-based multi-image deconvolution for removing handshake blur. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014).
13. Perrone, D., and Favaro, P. Total variation blind deconvolution: The devil is in the details. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014).
14. Sörös, G., and Flörkemeier, C. Blur-resistant joint 1D and 2D barcode localization for smartphones. In *ACM 12th International Conference on Mobile and Ubiquitous Multimedia (MUM)* (2013).
15. Sun, L., Cho, S., Wang, J., and Hays, J. Edge-based blur kernel estimation using patch priors. In *IEEE International Conference on Computational Photography (ICCP)* (2013).
16. Tai, Y.-W., Chen, X., Kim, S., Kim, S. J., Li, F., Yang, J., Yu, J., Matsushita, Y., and Brown, M. Nonlinear camera response functions and image deblurring: Theoretical analysis and practice. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 10 (2013).
17. van Gennip, Y., Athavale, P., Gilles, J., and Choksi, R. A regularization approach to blind deblurring and denoising of qr barcodes. *arXiv:1410.6333* (2014).
18. Whyte, O., Sivic, J., Zisserman, A., and Ponce, J. Non-uniform deblurring for shaken images. *International Journal of Computer Vision 98*, 2 (2012).
19. Xu, L., and Jia, J. Two-phase kernel estimation for robust motion deblurring. In *European Conference on Computer Vision (ECCV)* (2010).
20. Xu, L., Zheng, S., and Jia, J. Unnatural L0 sparse representation for natural image deblurring. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013).
21. Xu, W., and McCloskey, S. 2D Barcode localization and motion deblurring using a flutter shutter camera. In *IEEE Workshop on Applications of Computer Vision (WACV)* (2011).
22. Yahyanejad, S., and Ström, J. Removing motion blur from barcode images. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2010).
23. Zhang, Q., Xu, L., and Jia, J. 100+ times faster weighted median filter (WMF). In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014).