QfaR: Location-Guided Scanning of Visual Codes from Long Distances

Sizhuo Ma
sma@snapchat.com
Snap Inc.
New York City, NY, USA

Jian Wang
jwang4@snapchat.com
Snap Inc.
New York City, NY, USA

Wenzheng Chen
wenzheng@cs.toronto.edu
University of Toronto
Toronto, Ontario, Canada

Suman Banerjee
suman@cs.wisc.edu
University of Wisconsin-Madison
Madison, WI, USA

Mohit Gupta
mohitg@cs.wisc.edu
University of Wisconsin-Madison
Madison, WI, USA

Shree Nayar
snayar@snapchat.com
Snap Inc.
New York City, NY, USA

ABSTRACT

Visual codes such as QR codes provide a low-cost and convenient communication channel between physical objects and mobile devices, but typically operate when the code and the device are in close physical proximity. We propose a system, called QfaR, which enables mobile devices to scan visual codes across long distances even where the image resolution of the visual codes is extremely low. QfaR is based on *location-guided code scanning*, where we utilize a crowdsourced database of physical locations of codes. Our key observation is that if the approximate location of the codes and the user is known, the space of possible codes can be dramatically pruned down. Then, even if every "single bit" from the low-resolution code cannot be recovered, QfaR can still identify the visual code from the pruned list with high probability. By applying computer vision techniques, QfaR is also robust against challenging imaging conditions, such as tilt, motion blur,*etc.* Experimental results with common iOS and Android devices show that QfaR can significantly enhance distances at which codes can be scanned, *e.g.*, 3.6cm-sized codes can be scanned at a distance of 7.5 meters, and 0.5m-sized codes at about 100 meters. QfaR has many potential applications, and beyond our diverse experiments, we also conduct a simple case study on its use for efficiently scanning QR code-based badges to estimate event attendance.

ACM MobiCom '23, October 2–6, 2023, Madrid, Spain

ACM ISBN 978-1-4503-9990-6/23/10…$15.00
https://doi.org/10.1145/3570361.3592497

CCS CONCEPTS

• **Human-centered computing** → **Smartphones**.

KEYWORDS

visual codes, QR codes, location-guided, GPS, long distance, location-based services, mobile computing

ACM Reference Format:
Sizhuo Ma, Jian Wang, Wenzheng Chen, Suman Banerjee, Mohit Gupta, and Shree Nayar. 2023. QfaR: Location-Guided Scanning of Visual Codes from Long Distances. In *The 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '23), October 2–6, 2023, Madrid, Spain.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3570361.3592497

1 INTRODUCTION

Visual codes such as Quick Response (QR) codes create a bridge between the real world and the digital world. QR codes were invented back in 1994, but did not grow in popularity until smartphones became ubiquitous. A user, today, can simply point their phone to a QR code on the storefront of a restaurant to access a digital menu and a virtual checkout counter. In museums, QR codes augment real exhibits by providing access to digital multimedia content such as audio guides and video clips, all with the help of smartphones.

Despite their versatility and simplicity, QR codes require the users to scan the code from a close distance (<1m for centimeter-sized codes) for reliable decoding. If this requirement of close proximity could be removed, many new opportunities may emerge for this technology, making them more convenient and appealing to users, thereby increasing their deployability and impact. Imagine walking in Manhattan and spotting a QR code across the street. Wouldn't it be nice if you could just pull out your phone to scan it without having to cross the street (Fig. 1)?

Figure 1: Long-distance code scanning with QfaR. Conventional code scanning (WeChat QR [3]) only works for moderate distances (red boxes) and fails to decode heavily pixelated codes (red crosses). We propose a location-guided scanning method which uses *approximate* user location (*e.g.*, from GPS) and location of fixed codes to prune down the list of possible codes. QfaR can be used to scan codes at longer distances (green boxes, 4x conventional), which makes code scanning more effortless. Results are real outputs from QfaR, captured with an iPhone 12 Pro.

In this paper, we introduce QfaR, a technique that enables scanning of visual codes from a longer distance using common mobile cameras. [1] QR codes are usually packed with a large number of data bits encoded (> 100) to index a large number of entities. When scanned from a long distance, the captured images of the codes suffer from strong pixelation and other artifacts, making the bits indiscernible. This raises the following question: Do we need to recover every single bit in order to recognize the code in front of us?

Our key observation is that if the approximate geographical location of the user is known (*e.g.*, via GPS or other localization techniques) when they scan a code, the list of possible codes can be significantly *pruned down*. For example, in an outdoor urban environment, if there are 10,000 different codes within a radius of 200 meters (which may be quite typical in many common settings as discussed in Sec. 6.2), a lower-resolution image of the code captured from a long distance contains sufficient visual information to uniquely identify the code from the pruned list of codes with high probability, even if we cannot recover every bit in the original code (Fig. 3). This framework allows scanning QR codes from longer distances than other practical methods – our experiments demonstrate that QfaR can increase the scanning distance of QR codes by a *factor of 4 or more*, which can significantly enhance the type of applications and use cases. QfaR can be used at distances even where each dot of the code (one dot corresponds to one bit) occupies less than 1 pixel in the camera image, making it nearly impossible for conventional code readers to identify the code due to strong blur and pixelation artifacts (Fig. 1).

Prior efforts, such as PixNet [21], Strata [11] and FOCUS [10] have addressed similar problems (between cameras and displays) in different contexts, but require specialized visual codes for long range communication. In contrast, QfaR works with *existing QR codes*, thus requiring no new hardware or encoding infrastructure. QfaR also leverages crowdsourced location information of such codes, which is not considered in these prior efforts.

[1] While we only focus on QR codes as an example to demonstrate the proposed method, it is in principle broadly applicable to a general class of visual codes including 1D Barcode, Data Matrix, and Aztec codes.

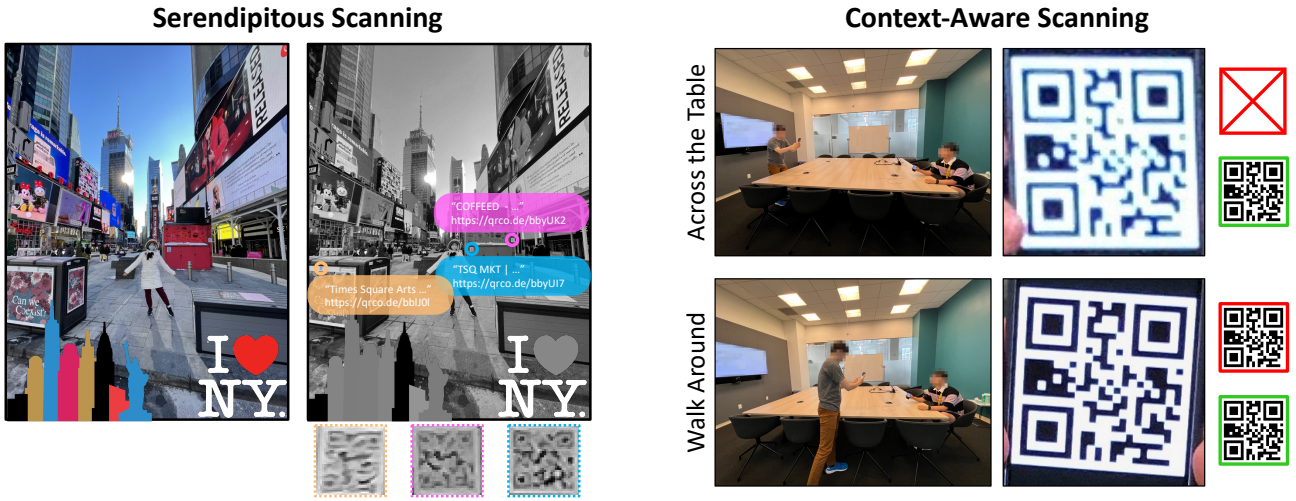

Figure 2: Additional use cases of QfaR. (Left) Besides being a robust QR scanner for *attentive* scanning, QfaR also enables *serendipitous discovery of codes*, which provides new opportunities for QR codes deployers to deliver digital contents, *e.g.*, in AR. (Right) Besides pruning code space using fixed code locations, QfaR can also be applied to pruning using contextual information for moving codes, *e.g.*, friending for social media apps. Users can friend other people from long distances, which reduces user friction and improves overall user experience. All results are real outputs from QfaR, captured with an iPhone 12 Pro (Left) and a Galaxy S22 (Right).

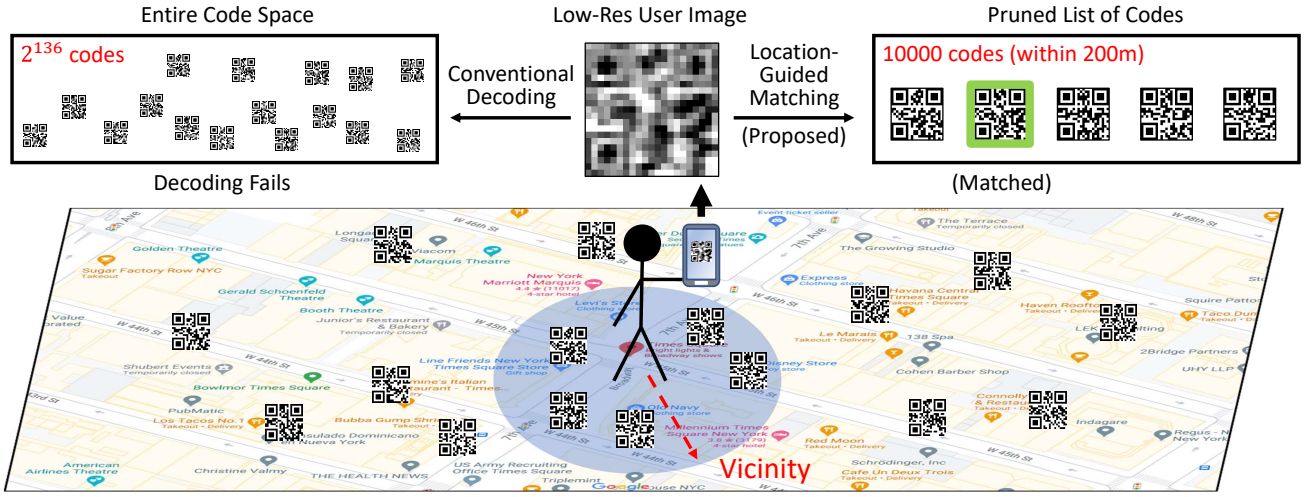

Figure 3: Location-guided scanning of visual codes. Given a low-resolution user image of the code, conventional decoding cannot find the right code from the huge entire code space (*e.g.*, 2^{136} codes). In the proposed location-guided scheme, approximate user location is used to prune down the list of possible codes given the fact that the user can only scan a code in their spatial vicinity (*e.g.*, 10000 codes within 200m). The scanned code image is then matched against the pruned list of codes to find the correct code. (Map data: Google Maps)

Location-guided decoding: The *location-guided pruning* of QfaR requires codes to have fixed and known approximate locations. A key challenge is to register the spatial locations of all such codes. One approach is to have the deployers of the codes manually register their locations, which is cumbersome and not scalable. Instead, we propose a *crowdsourcing* approach, where the locations of the codes are automatically determined from previous conventional code scans (called scanning a *full code*), without any extra effort from the users. The key idea is to infer code locations from user locations, as shown in Fig. 4. Since user location information can be noisy, the estimated code location may not be accurate initially,

Figure 4: Code localization. Approximate locations of users who successfully scanned a full code are used to estimate the location of the code. The estimate becomes more and more accurate as more users scan the code.

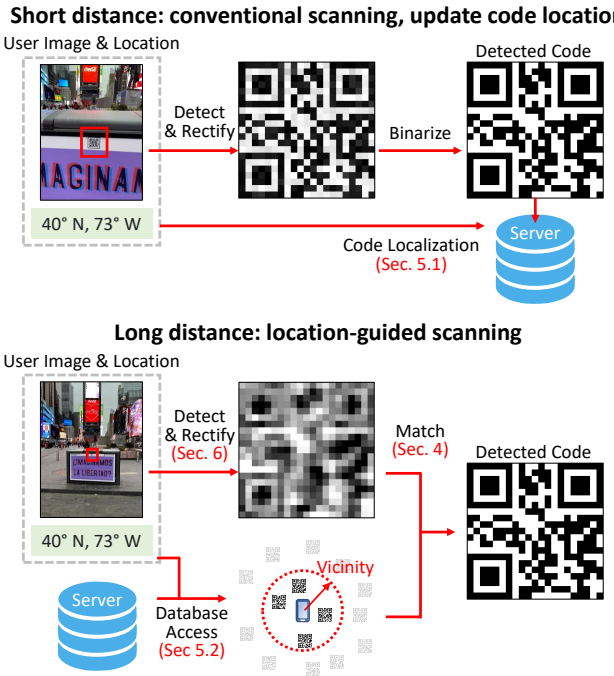

Figure 5: QfaR pipeline. (Top) Conventional scanning is used at short distances. User location is used to estimate the location of the code. (Bottom) Proposed location-guided scanning is used at long distances.

but the precision improves over time as more users scan the code. Estimated code locations are stored using efficient data structures on a server that allows for real-time access.

1.1 Application Scenarios for QfaR

What does the long-distance capability of QfaR imply in various application scenarios? Here we propose three typical use cases of QfaR.

Robust, easy scanning of existing static QR codes. The advantage of being able to scan QR codes at longer distances may be unclear at first glance as existing QR codes are mostly intended for close-range interactions, *e.g.*, scanning a menu. It is important to notice that, no matter how far a code is intended to be scanned, QfaR increases that distance by 4x or more. This is because the limiting factor for a QR code to be successfully scanned is how many pixels a single module (a black/white bit) occupies in the captured image. In Fig. 1(a), we show a long-range example where the scanning distance of a 0.5m-sized code is extended from 21m (conventional) to 88m (QfaR). Even for a short-range example such as scanning an acrylic menu stand in a restaurant, QfaR could extend the scanning distance from 0.5m to 2m, which allows easy scanning even across the table.

In addition to extended distances, location-guided pruning also provides robustness against image distortions due to other kinds of non-ideal imaging conditions, including large tilt angles (perspective distortion), motion blur, and low-light. To summarize, QfaR makes the scanning of existing QR codes with fixed locations more robust and effortless.

Serendipitous scanning. Beyond high-performance *attentive code scanning*, QfaR may *serendipitously* find a code even before the user notices it. Fig. 2 (Left) shows a concept image: The camera app picks up three tiny codes in the photo without being told explicitly to scan a code and displays the decoded information on top of the image. This *serendipitous scanning* of codes provides new opportunities for QR codes deployers to deliver digital contents. One such example is *augmented storefronts*: When a user sees through a smartphone screen or smartglasses, all shops display their bestselling products as augmented reality elements, providing a new form of shopping experience.

Context-aware code scanning: QfaR can also be used for non-static codes where the list of codes can be pruned down using contextual information instead of geographical information. Fig. 2 (Right) demonstrates a scenario where QfaR is applied to friending people on a social media app. Instead of building a database through crowdsourcing, users temporarily share their locations with the server when the friending mode is on. The code list is pruned down to the set of active users in the spatial vicinity, which can be quite small even when a low-precision user location is provided (*e.g.*, from IP address of the mobile network or Wi-Fi hotspot). QfaR then allows users to friend people across a larger distance (e.g., across the table without the need to walk around the table), which results in a smooth user interaction experience especially when there is a large group of people.

As another example, we conduct a case study where QfaR is utilized to take attendance in a large classroom. Each student wears a QR-badge, and regular mobile phones are used to take images to identify individuals present in the room. The study exposes various opportunities and challenges.

1.2 Scope and Implications

The main contribution of this work is a novel location-guided scanning framework, which enables robust, long-distance scanning, using only off-the-shelf and computationally efficient (Sec. 6) techniques such as template matching [4] and object detection neural networks [22], which are computationally efficient.

The proposed framework is not meant to replace existing visual code readers which can decode without communicating with a server. Instead, it is designed to complement existing readers and provide a smoother user experience. In practice, the code reader should switch seamlessly between the two modes, as shown in Fig. 5. The conventional decoding scheme is first used to scan the full code and update the estimated code location on the server if scan succeeds. If scan fails under challenging conditions, the reader automatically switches to the proposed location-guided scanning. The whole process is completely transparent to the user.

Following are the main contributions of this paper:

- We propose a novel location-guided decoding approach that is robust to channel impairments resulting from long distances and other artifacts.
- We derive an upper bound of the probability of decoding failure. We show that in practical scenarios, the probability of decoding failure is vanishingly low.
- We present a method for automatically building a spatially-indexed database of codes via crowdsourcing to support location-guided pruning.
- We develop a neural network-based code detector for robustly locating low-resolution codes in images.
- We empirically evaluate the decoding performance over a wide range of scenarios both in simulation and real experiments using common iOS and Android platforms. QfaR significantly increases the range and applicability of visual codes.

2 RELATED WORK

Image processing techniques for improving decoding performance. One idea to allow scanning visual codes at longer distances is to apply image super-resolution to the captured images before decoding. Super-resolution methods specifically designed for barcodes [14] and QR codes [15] have been proposed which perform better than conventional super-resolution methods as they take advantage of the traits of QR codes such as bimodal distribution of image intensities and grid-like spatial structure. Another idea to improve decoding performance is to rely on deep learning to robustly detect, align, and decode QR codes [20], or use a super-resolution network prior to decoding (WeChat QR [25]), which achieve state-of-the-art performance. These methods decode QR codes by extracting information from images only, while the proposed method can further improve decoding performance by utilizing location information.

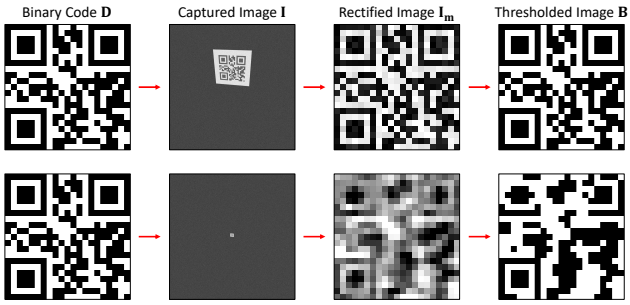

Figure 6: Notations for the conventional QR codes scanning. A QR code can be represented by a binary square matrix D, which appears as a quadrilateral in the captured image I. The code image is then rectified (I_m) and thresholded (B). Conventional Reed-Solomon decoding algorithms are applied to obtain the data payload (Top), but fail when the distance is long and B contains a lot of errors (Bottom).

Visual code designs for scanning from a long distance. Novel visual codes that are specifically designed for scanning from a wide range of distances have also been proposed. A class of spatially subdividing methods [2, 5, 11, 12, 24] divide the black or white dot in the base level into smaller blocks to encode information for the next level. Frequency domain approaches [10, 21] encode information in different bands in the frequency domain, and then converted to a 2D code in the spatial domain through inverse Fourier transform. Both space and frequency domain methods focus on novel multi-resolution code designs such that different bands of information are sorted by their importance such that the most important information is still decodable at a long distance.. Our work proposes a scanning framework that extends the maximum scanning distance of codes and is more general in the sense that it is compatible with all existing codes (*e.g.*, QR codes) and do not require replacing them with new codes. It is possible to combine both approaches in practice.

3 CONVENTIONAL QR CODE SCANNER

3.1 Conventional Detection and Decoding

Before introducing the proposed QfaR framework, we first give an overview on conventional QR code scanners. A QR code can be represented by a $n \times n$ binary square matrix $\mathbf{D} \in \{0,1\}^{n \times n}$. The resolution n is defined by the *version* of QR codes (*e.g.*, $n = 21$ for Version 1 QR codes). When a user take an image $\mathbf{I} \in [0,1]^{p \times q}$ ($p \times q$ is the camera resolution) of the code, the code appears as a quadrilateral whose shape is defined by the relative position and orientation of the code.

Conventional QR reader detects the fiducials at the corners, which is then used to warp back and resample the image into a *rectified image* $\mathbf{I}_m \in [0,1]^{n\times n}$ that has the same resolution as the code. Notice that $\mathbf{I}$ and $\mathbf{I}_m$ take continuous values between 0 and 1 because of pixelation, image noise as well as other nonlinear transforms during post-processing (See Sec. 4). A proper threshold $p \in [0,1]$ is then chosen to binarize the code $\mathbf{B}$: $B_{ij} = 1$ if $I_{ij} > p$ else 0. The entire imaging and processing pipeline is summarized in Fig. 6 (Top).

Ideally $\mathbf{B}$ should look exactly as $\mathbf{D}$. In practice, some bits of $\mathbf{B}$ can be erroneous due to image quality, code alignment error or incorrect binarization. To achieve robustness against a small number of error bits, the data bits on a QR code are arranged in groups of 8 bits (called a *symbol*) and encoded into Reed-Solomon codes for error correction [23].

3.2 Pruning in the Code Space

The detected black and white bits are rearranged into a Reed-Solomon codeword (called *received code* from now on), which contains errors due to image noise, alignment error, *etc.* Ideally we would like to find the codeword that is the closest to the received code since it is the *most likely* to be the correct codeword. However, in practice, there is no efficient way to identify the closest codeword: Traversing the entire space of codewords has an exponential complexity in terms of the code length. Instead, conventional QR code decoders use standard Reed-Solomon decoding algorithms (*e.g.*, Berlekamp-Massey algorithm [18]). Such algorithms are called *unique decoding* methods because the closest codeword can be uniquely recovered with very low computational complexity *if* the number of error symbols is less than $(1-R)/2$ (R is the information rate, the ratio between the amount of actual information and the amount of transmitted data). However, this distance bound limits the capability of scanning a QR code from a long distance (Fig. 6 (Bottom)).

A different decoding scheme called *list decoding* was later invented, which relaxed this distance bound by allowing the algorithm to return the list of all codewords within some distance d from the received code. In other words, the list of possible codes is *pruned* according to the Hamming distance in the code space. If the list contains multiple words, the decoder can go through the list and return the closest codeword if there is a unique closest one, or use semantic context or side information [8] to narrow down the right code. The most representative list decoding algorithm for Reed-Solomon codes is the Guruswami-Sudan algorithm [9] which can correct up to $1-\sqrt{R}$ errors.

However, the relaxation from $(1-R)/2$ to $1-\sqrt{R}$ is not significant for Reed-Solomon codes with a high rate R. For QR codes with error correction level L, this error correction radius is improved from 7% to 7.3% only. Because of the limited improvement and higher computational complexity, list decoding is not used in existing QR decoders.

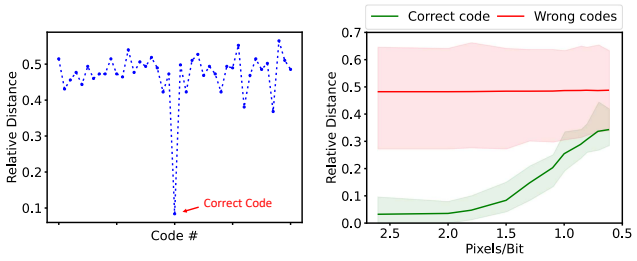

Figure 7: Relative distance between received code and matched codes as a function of code resolution. (Left) At high resolution, the distance from the received code to the correct codes is much lower than that to wrong codes. (Right) We plot the range and mean of distances to the correct code and wrong codes respectively. The two ranges are completely separated at high resolutions and overlap by only a small fraction at low resolutions such as 0.6×0.6 pixels/bit.

4 LOCATION-GUIDED DECODING

4.1 Decoding Through Binary Matching

As we discussed above, *pruning in the code space* does not improve the error correction capability significantly for QR codes. The question is, can we use other contextual information to prune down the list of possible codes for better error correction? We propose location-guided decoding, which can be viewed as *pruning in the space of geographic locations*. The list of possible codes is first pruned down using geographic distances, and then Hamming distance is used to find the (most likely) correct code. This *binary matching* method can be formalized as following:

$$\mathbf{D}_m = \arg\min_{\mathbf{D}\in S} d_{\mathrm{H}}(\mathbf{B},\mathbf{D}), \tag{1}$$

where $\mathbf{B}$ is the thresholded code (See Fig. 6). $\mathbf{D}$ denotes each of the candidate codes in the pruned list S. $\mathbf{D}_m$ is the match result. $d_{\mathrm{H}}(\cdot,\cdot)$ measures the Hamming distance between two codes (in terms of *bits* instead of Reed-Solomon symbols). Pruning in the geographic space is efficient even when there are a lot of errors. For example, in most cases there are no more than 10000 codes within the GPS uncertainty of the user. Brute-force searching in the pruned list is then possible.

Fig. 7 (Left) visually shows how the location-guided decoding works. When the user takes a high-resolution image of the code (short distance), the correct code can be easily identified by finding the code with shortest distance to the received code. To see how the gap between correct code and wrong codes narrows as the distance increases, we take

images of 85 different codes at a range of distances, match them against a database of 10000 codes and plot the range and mean of distances in Fig. 7 (Right). At high resolutions, the two ranges are perfectly separated and decoding always succeeds. At low resolutions, there is an overlap between the two ranges. However, the mean distance to correct codes is still below the minimum distance to wrong codes in most cases, which means that there is still a chance for the decoding algorithm to return the correct word. Notice that this error-correction capability is independent of the error correction level of the QR codes. For a level L QR codes, it is still possible to correct 30% of error bits using binary matching.

Decoding capability. What is the success rate of location-guided decoding? We derive an upper bound for failure probability to decode binary codes without error correction, which also provides insights for other visual codes:

Theorem. *Given a received binary code of bit length n (no error correction), if the number of wrong bits is d, the number of codes in geographic vicinity is m, then the probability of failing to find the correct code in the list has an upper bound of* $\frac{mf(n,d)}{2^n-m}$, *where* $f(n,d) = \sum_{i=0}^{d} \binom{n}{i} - 1$.

Proof. The total number of wrong codes that are closer to the received code (in terms of bits) than the correct code is given by $f(n,d) = \sum_{i=0}^{d} \binom{n}{i} - 1$. The probability of matching to the correct code can then be expressed by the probability of randomly choosing m codes from 2^n codes which are all farther away from the received code than the correct code,

$$\begin{aligned} p_{\text{success}} &= \binom{2^n - f(n,d)}{m} \Big/ \binom{2^n}{m} \\ &= \frac{\prod_{j=0}^{m-1} 2^n - f(n,d) - j}{\prod_{j=0}^{m-1} 2^n - j} \\ &> \left(\frac{(2^n - f(n,d) - m)}{2^n - m}\right)^m \\ &= \left(1 - \frac{f(n,d)}{2^n - m}\right)^m \\ &\geq 1 - \frac{mf(n,d)}{2^n - m} \qquad \text{(Bernoulli's inequality).} \end{aligned} \tag{2}$$

The last line assumes $\frac{f(n,d)}{2^n-m} < 1$, which holds if $d < n/2$ (less than half of the bits are wrong) and $m < 2^{n-1}$ (less than half of the codes are in the vicinity). The probability of failure is,

$$p_{\text{failure}} = 1 - p_{\text{success}} \leq \frac{mf(n,d)}{2^n - m}. \qquad \square \tag{3}$$

To see what this bound implies in practice, let us consider a 400-bit code. Fig. 8 (Left) shows the bound as a function of number of error bits. As there are more errors, decoding is more likely to fail, but the probability of failure is negligible when the error bits are less than 35%, even when there are 1 million codes in vicinity. To study the relationship between the error-correction capability and the number of codes, Fig. 8 (Right) plots contours in the probability bound space: To achieve an upper bound of 1% probability, how much error bits can be tolerated given the number of codes in vicinity? As shown in the plot, the ratio of error bits decreases as the number of codes increases. Another interesting fact is that for the same number of codes, longer codes are tolerable to higher ratio of errors.

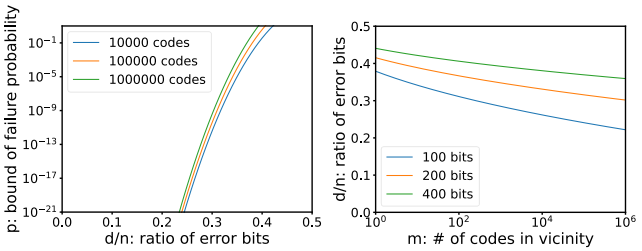

Figure 8: Probability of failure to decode. (Left) The upper bound of probability increases as the number of error bits increases and the number of codes increases. (Right) To ensure the probability of error is below 1%, given number of codes in vicinity, we plot the maximum fraction of errors that is allowed.

4.2 Intensity-Based Matching

One limitation of binary matching is the loss of information when thresholding the intensity values. For example, a pixel with an intensity of 0.9 is more likely to be a white bit than a pixel at 0.7, which is lost after thresholding. Therefore, the intensity image $\mathbf{I}_m$ contains more information for distinguishing the correct code. We propose *intensity-based matching* which finds the code $\mathbf{D}_m$ with the shortest L^2 distance to the rectified intensity image $\mathbf{I}_m$:

$$\mathbf{D}_m = \arg\min_{\mathbf{D} \in S} d_{L^2}(\mathbf{I}_m, \mathbf{D}). \tag{4}$$

Fig. 9 compares the performance of binary matching and intensity-based matching on simulated code images, assuming the detector and aligner work perfectly. For different code resolutions (distances) and tilt angles, intensity-based matching performs consistently better. We focus on intensity-based matching for the rest of the paper.

Relation to soft decoding. The proposed intensity matching can be seen as a *soft decoding* approach [16], which is an extension of list decoding. In soft decoding, at each location of the codeword, each possible symbol is assigned with a probability. Here, the probability of each bit being 0 or 1 is implicitly measured by the L^2 distance.

Robustness to poor imaging conditions. This approach is also reminiscent of template matching [4] in computer vision.

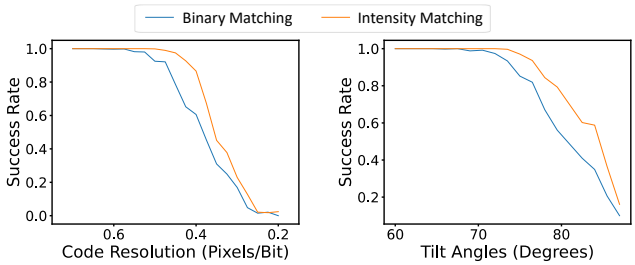

Figure 9: Comparison between binary matching and intensity-based matching. For different code resolutions (distances) and tilt angles, the performance of intensity-based matching is consistently better than binary matching because the probability information carried by the intensities is taken into account.

The rectified image $\mathbf{I}_m$ can be seen as a corrupted version of the true binary code $\mathbf{D}$ under nonidealities including camera noise, pixelation, motion blur, *etc.* The proposed intensity matching is robust against such degradation because the low-frequency image component is preserved. For example, a *blob* of black bits remains identifiable in the presence of pixelation, motion blur, or small misalignment.

Image pre-processing. So far, we have assumed that the captured image intensities are the same as real-scene intensities. In practice, that is not the case because the scene intensities are usually transformed by various nonlinear processes during imaging, which is often modelled as a nonlinear function called a camera response function [7]. To mitigate the bias introduced in this nonlinear process, we propose the following simple algorithm:

- Map the percentiles (5%, 95%) to intensity range [0, 1] using a linear mapping (identify intensity range and exclude outliers).
- Then map the median to 0.5 using a gamma function $y = x^\gamma$ (compensate for the nonlinearity).

Confidence of matching. To evaluate how reliable the matching result is, we propose a confidence measure: For a received code $\mathbf{B}$, we find the first two matches $\mathbf{D}_1$, $\mathbf{D}_2$ with shortest distances. The confidence measure is defined as:

$$c = 1 - \frac{d(\mathbf{B}, \mathbf{D}_1)}{d(\mathbf{B}, \mathbf{D}_2)} \tag{5}$$

When c is small, the matching result is unreliable, and the system can display the top n matches for the user to choose from if the target application is not security-sensitive. If security is important, we report a failure when $c < c_{\text{th}}$, where c_{th} is a predetermined threshold. The choice of c_{th} determines the precision and recall of the system, which should be determined according to the desired reliability of the application. In our prototype system we use $c = 0.03$.

5 CODE LOCALIZATION

To facilitate location-guided pruning, we need to build a database of location-registered codes. For a large-scale ecosystem with millions of codes available, it is infeasible to build such a database manually. Therefore, we propose an approach to build such databases through *crowdsourcing*, *i.e.*, automatically from previous successful user scans of *full codes* to bootstrap the location-guided pruning framework. This process is completely transparent and effortless for the users: As more and more users scan the codes, the locations of the codes are estimated and registered in the database, with the uncertainties of the estimates decreasing over time. The system then seamlessly switches to location-guided pruning to allow code scanning at a longer range. The problem is formalized as following:

Given: the 3D coordinates of n users: (u_x^i, u_y^i, u_z^i) for $i = 1 \ldots n$ (usually comes from GPS),
Find: the 3D coordinates of the code: (p_x, p_y, p_z).

We adopt a simple approach to estimate code location by taking the average GPS location of the users who have scanned the code. Since most smartphones also provide an estimate of GPS accuracy, the code location can be estimated by weighted least squares:

$$\min_{p_\square} \sum_i \frac{(p_\square - u_\square^i)^2}{{\sigma_\square^i}^2}, \ \square = x, y, z\,. \tag{6}$$

GPS errors in the three dimensions are assumed to be uncorrelated, zero-mean Gaussians with the standard deviation $(\sigma_x^i, \sigma_y^i, \sigma_z^i)$, which depends on GPS signal strength and interference and in general varies for different user locations. In an urban scene, σ_x^i and σ_x^i can vary from 2m to 12m depending on signal strength and interference (calculated from [19]). For example, Android phones return the radius of 68% confidence r as the horizontal location accuracy. If we assume $\sigma_x = \sigma_y$, then σ_x can be converted from r: $\sigma_x = 0.662r$.

Note that this approach has a inherent bias in the estimated code location because people can only scan the code from its front, not from the back. Therefore, the estimated location will always be slightly biased towards the front-facing direction of the code. The impact of this bias can be minimized by setting a conservative estimate of the code location uncertainty (Sec. 6.2).

Identification of fixed codes. The basic assumption behind location-guided pruning is that every code in the database has a fixed location. This is ensured by only including codes that have been scanned for a few times at approximately the same location. Furthermore, a timestamp is attached to each code such that when a code has not been scanned for a period of time (*e.g.*, 30 days), we conclude the code may have been removed and delete it from the database.

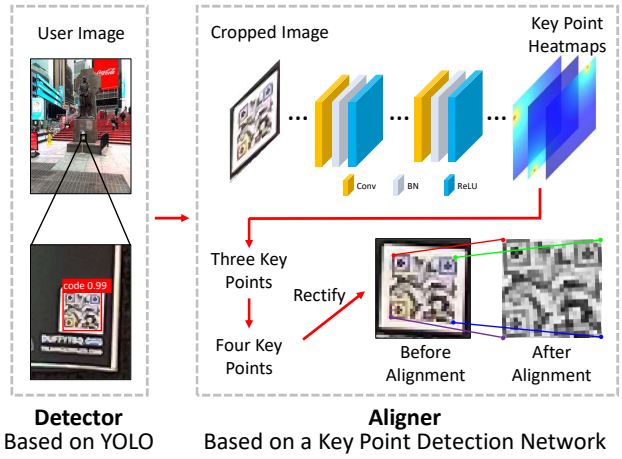

Figure 10: A YOLO-based detector detects bounding boxes of codes in the user image. The bounding boxes are cropped and passed to an aligner network which finds three corners (upper-left, upper-right, and bottom-left). The fourth corner is calculated assuming the code being a parallelogram. Geometric transform is applied to rectify the code, which is then decoded using the proposed soft decoding algorithm.

6 COMPUTATIONAL REQUIREMENTS

As a mobile computing system, QfaR consists of a mobile scanner app and a decoding service running on a server. The mobile app takes a picture, detects the codes and aligns them into rectified code images, which are small (about 1kB) and can be transmitted to the server with small bandwidths requirement. The server performs decoding and send the results back. In this section, we discuss the computational requirements on both the mobile side and the server side.

6.1 Mobile Side: Code Detection

Traditional QR code detection algorithms rely heavily on identifying the fiducial markers at the corners, which consist of fixed black-white intervals. As a result, they are not resistant to low resolution, large tilt angles, motion blur, *etc.* Inspired by recent success of deep learning in object detection, we propose a neural network-based code detector and aligner (see Fig. 10) that can detect codes even at extremely low resolutions (0.5×0.5 pixels/bit) where the fiducials are barely identifiable even by human eyes.

For the detector, we train a state-of-the-art object detection network YOLO [22] on QR code data to obtain bounding boxes of codes in an image. To generate sufficiently large amounts of data for training, we write a QR code image simulator that carefully models the imaging pipeline of common smartphone cameras. The gap between synthesized data and real data captured by smartphones is small because of the structural simplicity of QR codes.

The bounding box is used to crop the image, with extra margin to reduce incomplete cropping, which is then passed to an aligner network. The aligner employs a 7-layer convolutional neural network to regress the cropped code region (resized to 128×128) into three 64×64 heat maps. Each heat map represents the probability of the corresponding corner. The corner location is the sum of the probability-weighted pixel locations. The network outputs three corners where the fiducials are located, while the fourth corner is calculated by assuming the code is a parallelogram, which holds when the code is small compared to the scanning distance.

Our prototype detector and aligner currently runs on a PC, but they can be readily ported to mobile phones. Various lightweight variants of YOLO [1, 13] detector have been proposed, which can be trained with our QR code training images and run efficiently on smartphones. We have implemented our aligner network on Android phones, which only occupies 500kB memory and runs at 40-60ms per image on a Samsung S10e smartphone.

Does QfaR outperform human eyes at code detection? Studies have shown that the maximum angular resolution of human eyes, in the central fovea, is 28 arc seconds [6]. In contrast, a 12-megapixel camera with a regular lens (29 mm equivalent focal length) has an angular resolution of 63 arc seconds. Roughly speaking, eyes can see the code from the distance where the camera's sampling ratio is about $0.5/(63/28) = 0.24$ pixels / bit. With context information, eyes can probably see the code from even longer distances. Although currently QfaR has not achieved the limit of human eyes, it significantly narrows the gap.

Notice that although some cellphones on the market support over 20x zoom and may seemingly make QfaR surpass human vision, such zoom capability is achieved via digital zoom, which does not improve image quality for fine structural details such as QR codes.

6.2 Server Side: Location-Guided Decoding

To achieve efficient computation on the server side, codes must be organized in the database via *spatial indexing*. A widely-used data structure for spatial indexing is *R-tree*. For location-guided pruning, it is usually sufficient to use a 2D *grid*, because each query does not have to find exactly all the data entries within a radius; it is acceptable if the database returns a superset of entries as long as the set size does not degrades decoding performance significantly.

When a user scans a code, a search radius is computed by taking the sum of the uncertainty of user location, maximum scanning distance (*i.e.*, the longest distance at which the code

is still decodable empirically), and the maximum uncertainty of code location. The server then finds an approximate set of codes that reside within this search radius. For example, let us assume the uncertainty of user location is 20m (Sec. 5), the maximum distance a code is detectable at is 150m and the maximum uncertainty of code location is 30m (conservative estimate), which gives a search radius of 200m. We assume codes in a big city have a density of 10000 codes within 200m, which is an overestimate (on average there is a fixed QR code in every 2m×2m area)[2]. The pruned list then contains about 10000 codes. We use this rough estimate of pruned list size in our simulated experiments.

In the prototype system, we store 100 user past scans with each code for localization. For a city with 1 million codes, the database takes about 2GB of storage. Decoding with a list of 10000 codes can be done in 30ms on a laptop PC with our current unoptimized implementation, and can be further reduced by code optimization and running on a server.

Maintenance overhead. Although at first glance it may seem costly to deploy QfaR at city scale, one important observation is that we do not require the precise geospatial distribution of codes. In fact, we only need approximate code locations, which automatically emerges from previous scans and does not need to be updated in real-time. Therefore, it is sufficient to only update the database asynchronously, without affecting user experience.

7 EXPERIMENTAL RESULTS

7.1 Performance at Long Distances

In this section, we show that the long-distance scanning capability of QfaR is far beyond conventional approaches. *For 4cm-sized codes at 6m away, or 1m-sized codes at 150m away, QfaR can correctly decode them with 90% accuracy, which is more than 6x farther than conventional decoders.*

Accuracy of detector, aligner and decoder. To exhaustively evaluate the performance of each component in the decoding pipeline individually in a controlled manner, we physically model the camera imaging process and synthesize code images captured at different distances, angles, motion blur and lighting. For each set of parameters, we synthesize 100 images and compute the average accuracy. The simulated pruned code list contains 10000 randomly generated QR codes and the ground truth code. For other experiments in this paper, unless otherwise specified, we always assume a pruned list of 10000 codes, which is an overestimate of a densely populated city and is explained in Sec. 6.2 in detail.

[2]Note that it is possible to exceed this density in an exceptional, busy local area, *e.g.*, 100 codes within a radius of 10m. However, QfaR does not make any assumptions on this local code density. Its performance only depends on the number of codes within the search radius.

Fig. 11(a) shows that the detector achieves an accuracy of higher than 0.9 when the code resolution is higher than 0.5×0.5 pixels/bit. When the resolution goes lower, the detector can still detect the codes but also output false positives. Thus, we choose a threshold on the confidence values that limits detection of codes below 0.5 pixels/bit.

Fig. 11(b) plots the mean L^2 error of the detected corner locations. When testing the aligner and the decoder, we assume all prior components work perfectly. The mean error is lower than 1 bit as long as the code resolution is higher than 0.5×0.5 pixels/bit. To evaluate how this alignment error affects decoding performance, we manually perturb the ground truth corner location and plot the decoding success rate as a function of misalignment error in Fig. 11(c). Specifically, a Gaussian-distributed error is added to the x- and y- coordinates of the detected corners independently. The decoder is robust against small misalignment up to 1 bit, which means the accuracy of the aligner is sufficient.

Fig. 11(d) shows that the decoding success rate decreases as the number of codes increases and the code length decreases (*i.e.*, lower QR code version). Notice that for a pruned list of 10000 Version 1 code, the decoder achieves a success rate of 1.0 even at a resolution of 0.5×0.5 pixels/bit.

Comparison with conventional decoders. For the rest of the figure, we first show a quantitative plot of success rates on simulated data with full control of distance and various other parameters, then show qualitative results on challenging real images captured by iPhone 12 Pro and iPhone X. We compare QfaR with two baselines: OpenCV [3] and WeChat QR [25]. We notice that the OpenCV QR scanner based on conventional image processing completely fails the experiments. Therefore, we combine our proposed code detector and OpenCV decoder as our first baseline. We also compare with WeChat QR, which is the state-of-the-art open-source scanner and is an extra module of OpenCV. WeChat QR uses a similar, learning-based approach for code detection as QfaR (based on SSD [17]). However, it relies on unique decoding and is less robust against pixelation and other artifacts.

Robustness to long distances. The scanning performance depends on the size of the code in terms of pixels in the captured image, which is inversely proportional to the ratio of camera-code distance and the physical size of the code. To evaluate the performance, we plot the scan success rate against this ratio in Fig. 11(e). For example, if the code has a size of 1m, both conventional scanners achieve less than 90% success rate at 25m, while the proposed scanner achieves 90% at even 150m. This result is verified by real data in both Fig. 11(e) and Fig. 1, where QfaR can decode the codes at 4x (or more) longer distance than conventional scanners.

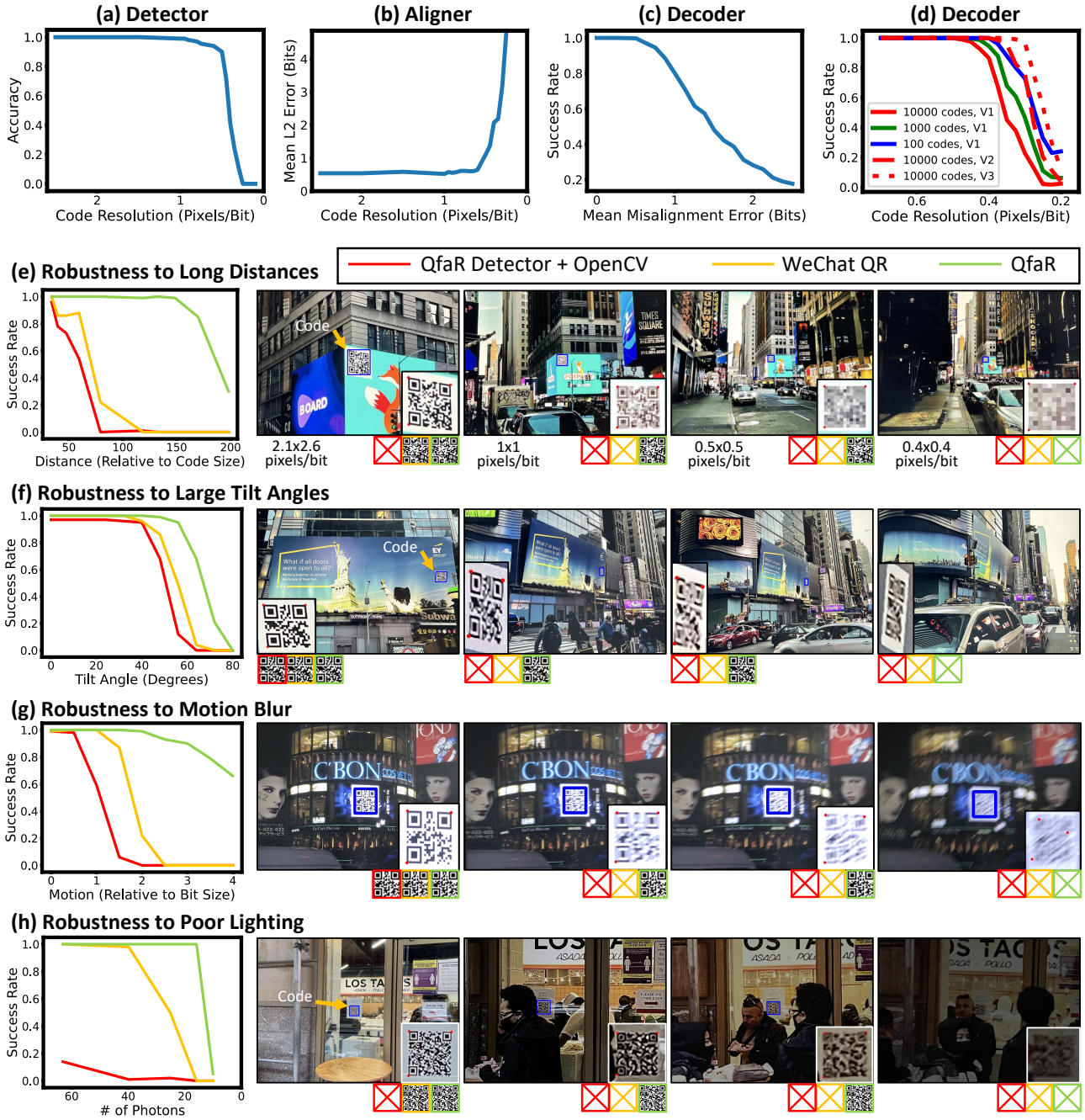

Figure 11: Experimental results. (a) The detector can detect small codes down to the resolution 0.5×0.5 pixels/bit. (b) The aligner can localize the code corners up to an error of 1 bit for codes larger than 0.5×0.5 pixels/bit. (c) The decoder is robust against small misalignment (less than 1 bit). (d) The decoder can find the correct code for codes larger than 0.5×0.5 pixels/bit. The success rate is higher if there are fewer codes in the list or the code length is longer. (e) Robustness to long distances. (f-h) Robustness to other challenging imaging conditions. In each row, we first show a plot of success rate on simulated data, and then show real images to verify the conclusion. We show a zoom-in view of the codes, where the estimated corners are plotted in red circles. Decoding results for the conventional and location-guided scanners are indicated in red, yellow and green windows (cross if not decoded). In all cases, the location-guided scanner is significantly more robust than the conventional approaches.

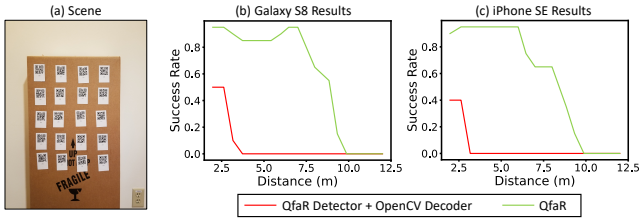

Figure 12: Controlled experiments for attendance tracking case study. 20 3.6cm×3.6cm QR codes are attached to a box and imaged by two phones at different distances. The conventional decoder fails to decode half of the codes at 2m, while QfaR can decode the majority of codes at 8m for both phones.

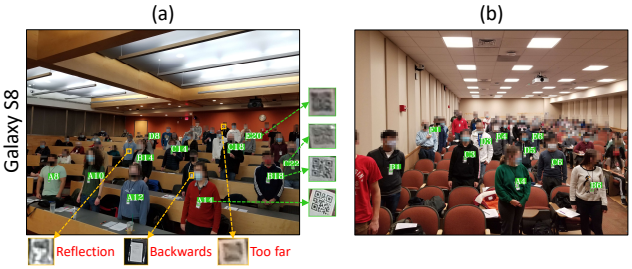

Figure 13: Case study: attendance tracking with context-guided pruning. Correctly decoded codes are shown in green labeled with seat numbers. Three failure modes are highlighted in yellow. Most codes within 7m can be decoded, which suggests QfaR can be a potential solution to automated attendance systems.

7.2 Challenging Imaging Conditions

Robustness to large tilt angles. Codes are more difficult to decode when they are tilted (not parallel to the image plane) because they are more challenging to detect and the effective resolution is decreased. Fig. 11(f) shows the scanning performance at different tilt angles, which is defined as the angle between the code plane and the image sensor plane. Conventional scanners achieve less than 90% success rate at 45 degrees, while QfaR achieves 90% up to 60 degrees. The real images are captured using iPhone 12 Pro in an outdoor scene at different angles.

Robustness to motion blur. Fig. 11(g) shows the performance at different amount of motion blur, which is measured by the amount of motion in the image space, relative to the size of a single black/white bit. For example, if each bit occupies a 5×5 pixels square, conventional approaches 90% success rate when the motion is 3.5 pixels and 7 pixels respectively, while QfaR achieves 90% up to 13-pixel motion. We display a photo on a monitor and quickly shake an iPhone X to capture the real images.

Robustness to poor lighting. Fig. 11(h) shows the performance at different levels of environmental lighting, which is measured by the number of photons received by the image sensors at each pixel. We also print out a photo and adjust the exposure of an iPhone 12 Pro to simulate images captured at different light levels. Although this does not reflect the real dynamic range of the scene, it serves our goal of evaluating the performance in extremely dark conditions. In such conditions, QfaR outperforms the two baselines.

7.3 Case Study: Attendance Tracking with Context-Guided Pruning

As discussed in Sec. 1, QfaR can be used not only for location-guided pruning, but also can be extended to specific scanning scenarios where the code list is pruned using contextual information. To help understand how QfaR can be used in a real-world code-scanning application, this section presents a case study of using QfaR as an automated attendance system. We use a specific implementation of QfaR where the code space is pruned using the context that only 300 QR codes (V1, error correction level L) are used in the attendance system, which are printed out at 3.6cm×3.6cm and distributed to the students as wearable badges. Before conducting experiments with the students, we perform a controlled experiment where we attach 20 of these badges on a flat surface and capture images from different distances using two different phones (Galaxy S8 and iPhone SE 2020) (Fig. 12). We notice that WeChat QR cannot handle challenging images that contain a large number of small codes next to each other, and fails to detect the codes even at 1.6m. Therefore, we only plot the decoding success rate of QfaR and QfaR detector + OpenCV decoder (See Sec.7.1) as a function of distance. The conventional decoder fails to decode half of the codes even at a short distance of 2m, while QfaR can decode the majority of codes at 8m for both phones.

We then conduct an experiment in multiple university classrooms. Each student was handed a QR-coded badge which encoded the seat number of the student, which established ground truth as we knew exactly which code should be located at which seat number. We took photos of the students with the two phones. The students were asked to stand up in groups and in each image we attempt to correctly identify all students standing.[3]. Fig. 13 shows the results of running QfaR on the Galaxy S8 photos and matching the

[3]Experiment approved by the Institutional Review Board, conducted with student content. Student names not retained, and faces are blurred at source to preserve privacy.

decoded code to seat numbers. Correctly detected and decoded codes are shown in green with the seat number labeled. We also show a few close-up views of the correct codes to demonstrate how the decoding process becomes challenging as the distance increases. Decoded codes with a low confidence are rejected and not shown. Since this is a challenging scenario with a large number of small codes in a cluttered environment, neither OpenCV nor WeChat QR is able to scan a single code. On the other hand, QfaR correctly decodes almost all codes within 7m, which suggest QfaR can be used as a robust, automated solution for attendance tracking. Three failure modes are highlighted in yellow, which are reflection, extremely long distance, and the fact that some students wore the badges backwards.

Some observations on the case study: It should be apparent that a single camera pointed at a single direction is not sufficient to provide sufficient coverage — due to occlusion and distance. In practice, multiple images from different camera positions and directions are still necessary to cover the entire classroom. Badges need to be double-sided for better performance. Camera type and quality also has an impact. For instance, fewer codes are decoded from iPhone SE images, which is probably due to a weaker camera and different processing steps by the image signal processors (ISP). A fully developed and robust attendance system based on QfaR with minimal engineering efforts.

8 LIMITATIONS AND DISCUSSION

Does QfaR work for any QR code? The proposed QfaR system can deal with various corner cases. For moving QR codes such as codes on a bus, QfaR will not put them in the database since they can be scanned at various different locations. We put a code in the database only if it has been scanned at the same locations for several times, as described in Sec 5. Another corner case is aesthetic QR codes which can come with different colors and even have logos embedded. Since QfaR treats QR codes as images and utilizes computer vision techniques to match them, aesthetic QR codes can also be correctly matched and decoded.

Although QfaR in principle can work for any static QR codes, not all applications will benefit from extended range. Scanning from a long distance may even be unfavorable for security-sensitive applications such as mobile payment (discussed below). Furthermore, QfaR is most beneficial to high-density locations like cities. Since a code needs to be scanned frequently to stay in the database, it may not stay long in the database for less populated areas, resulting in reduced benefit.

Security. While QfaR is widely applicable to various scenarios, there are limitations and considerations that require further thoughts and design. The serendipitous discovery capability allows detection of QR codes without users' attention, which is more susceptible to QR codes that embed malicious URLs. This can be prevented by not accessing the URL without users' consent. A related potential threat is that a private-domain code for authentication (*e.g.*, ticketing for entry) may be stored in the database and matched to a similar code being scanned. While we have proven that such scenario is unlikely to happen, we suggest using conventional code scanners for sensitive tasks with QfaR turned off.

Privacy. To use QfaR, users need to agree to share their GPS locations. We note that GPS sharing is becoming increasingly common on mobile apps; several apps (e.g., Uber, Yelp) work on the premise of sharing GPS, which suggests users' reluctance to share location may not be as severe as it appears. Nevertheless, it is important to disclose how location information is used and let the user determine if they want to share location anonymously to get a better code scanning experience.

Probabilistic code matching. Our analysis and experiments assume the codes are randomly generated. This is not true for all scenarios as the code deployer may generate codes with fixed structure, which is outside the purview of the code scanner. For example, typical URLs share the same starting string. Therefore, a slight drop in performance is expected when using QfaR as a generic QR scanner. For applications where the generation of codes can be controlled (*e.g.*, friending and attendance tracking), pseudorandom generation of codes should be used for best performance.

Network access. Since QfaR requires network access to the server during scanning, it is expected to take longer than conventional decoding, which may slightly reduce the responsiveness, of which the impact on user experience remains to be studied. Hence, we believe that location-guided scanning should not replace but complement conventional scanning: conventional scanning is still used at short distances, while location-guided scanning is only triggered at long distances. We hope to systematically explore these and other such issues in our future work.

ACKNOWLEDGMENTS

Suman Banerjee is supported in part through the following grants — US National Science Foundation's CNS-2112562, CNS-2107060, CNS-2003129, CNS-1838733, and CNS-1647152, and the US Department of Commerce's 70NANB21H043.

REFERENCES

[1] Pranav Adarsh, Pratibha Rathi, and Manoj Kumar. 2020. YOLO V3-Tiny: Object Detection and Recognition Using One Stage Improved Model. In *International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE, Coimbatore, India, 687–694.

[2] Eric P Batterman and Donald G Chandler. 1992. Multiple Resolution Machine Readable Symbols. US Patent 5,153,418.

[3] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[4] Roberto Brunelli. 2009. *Template Matching Techniques in Computer Vision: Theory and Practice.* Wiley Publishing.

[5] Ronald Steven Cok. 2013. Multi-Resolution Optical Codes. US Patent 8,439,275.

[6] Michael F Deering. 1998. The Limits of Human Vision. In *2nd International Immersive Projection Technology Workshop*, Vol. 2.

[7] M.D. Grossberg and S.K. Nayar. 2004. Modeling the Space of Camera Response Functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 10 (Oct. 2004), 1272–1282.

[8] V. Guruswami. 2003. List Decoding with Side Information. In *IEEE Annual Conference on Computational Complexity.* IEEE Comput. Soc, Aarhus, Denmark, 300–309.

[9] V. Guruswami and M. Sudan. 1998. Improved Decoding of Reed-Solomon and Algebraic-Geometric Codes. In *Proceedings 39th Annual Symposium on Foundations of Computer Science.* IEEE Comput. Soc, Palo Alto, CA, USA, 28–37.

[10] Frederik Hermans, Liam McNamara, Gábor Sörös, Christian Rohner, Thiemo Voigt, and Edith Ngai. 2016. FOCUS: Robust Visual Codes for Everyone. In *Annual International Conference on Mobile Systems, Applications, and Services (MobiSys).* ACM Press, Singapore, Singapore, 319–332.

[11] Wenjun Hu, Jingshu Mao, Zihui Huang, Yiqing Xue, Junfeng She, Kaigui Bian, and Guobin Shen. 2014. Strata: Layered Coding for Scalable Visual Communication. In *Annual International Conference on Mobile Computing and Networking (MobiCom).* ACM Press, Maui, Hawaii, USA, 79–90.

[12] Qingfeng Huang, James E Reich, Marc E Mosko, and Victoria ME Bellotti. 2014. Using Multi-Resolution Visual Codes to Facilitate Information Browsing in the Physical World. US Patent 8,849,943.

[13] Rachel Huang, Jonathan Pedoeem, and Cuixian Chen. 2018. YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers. In *IEEE International Conference on Big Data (Big Data).* IEEE, Seattle, WA, USA, 2503–2510.

[14] Renchao Jin, Shengrong Zhao, Xiangyang Xu, Enmin Song, and Chih-Cheng Hung. 2016. Super-Resolving Barcode Images with an Edge-Preserving Variational Bayesian Framework. *Journal of Electronic Imaging* 25, 3 (June 2016), 033016.

[15] Yuji Kato, Daisuke Deguchi, Tomokazu Takahashi, Ichiro Ide, and Hiroshi Murase. 2011. Low Resolution QR-Code Recognition by Applying Super-Resolution Using the Property of QR-Codes. In *International Conference on Document Analysis and Recognition (ICDAR).* IEEE, Beijing, China, 992–996.

[16] R. Koetter and A. Vardy. 2003. Algebraic Soft-Decision Decoding of Reed-Solomon Codes. *IEEE Transactions on Information Theory* 49, 11 (Nov. 2003), 2809–2825.

[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. 2016. SSD: Single Shot MultiBox Detector. In *European Conference on Computer Vision (ECCV).* 21–37. arXiv:1512.02325

[18] J. Massey. 1969. Shift-Register Synthesis and BCH Decoding. *IEEE Transactions on Information Theory* 15, 1 (Jan. 1969), 122–127.

[19] Krista Merry and Pete Bettinger. 2019. Smartphone GPS Accuracy Study in an Urban Environment. *PLOS ONE* 14, 7 (July 2019), e0219890.

[20] Shree K Nayar, Jian Wang, and Wenzheng Chen. 2022. Long Distance QR Code Decoding. US Patent 11,461,924.

[21] Samuel David Perli, Nabeel Ahmed, and Dina Katabi. 2010. PixNet: Interference-Free Wireless Links Using LCD-camera Pairs. In *Annual International Conference on Mobile Computing and Networking (MobiCom).* ACM Press, Chicago, Illinois, USA, 137.

[22] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. *arXiv:1804.02767 [cs]* (April 2018). arXiv:1804.02767 [cs]

[23] I. S. Reed and G. Solomon. 1960. Polynomial Codes Over Certain Finite Fields. *J. Soc. Indust. Appl. Math.* 8, 2 (June 1960), 300–304.

[24] Steven J Simske, Guy Adams, and Jason S Aronoff. 2010. Multiple Resolution Readable Color Array. US Patent 7,673,807.

[25] WeChat. 2021. WeChat QR code detector for detecting and parsing QR code. https://github.com/opencv/opencv_contrib/tree/4.x/modules/wechat_qrcode.